POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea Magistrale in
Ingegneria Meccanica

Progettazione e sviluppo di un sistema programmabile di Visione Artificiale e Realtà Aumentata a controllo e supporto di procedure di manipolazione pianificate

Relatore: Prof. Eugenio CASTELLI

Correlatori: Dott. Stefano MOTTURA
Prof. Marco TAUSEL

Tesi di Laurea di:
Alessandro LA ROSA
Matr. 755426

Anno Accademico 2011 – 2012

«The most exciting phrase to hear in science,
the one that heralds new discoveries,
is not "Eureka!" (I found it!) but "That's funny ..."»

Isaac Asimov (1920 - 1992)

Ringraziamenti

Un sincero ringraziamento al mio relatore, il prof. Eugenio Castelli, per la disponibilità, la cortesia ed i preziosi consigli.

Ringrazio il prof. Marco Tausel, per avermi dato la possibilità di svolgere lo stage presso ITIA–CNR e per il tempo dedicatomi durante il lavoro di tesi.

Un particolare ringraziamento al Dott. Stefano Mottura, paziente supervisore del mio lavoro quotidiano. La sua guida ed il suo sostegno sono stati per me un modello positivo di serio impegno nello studio e nella ricerca.

Ringrazio, inoltre, tutte le persone conosciute durante il periodo di Tesi all'interno di ITIA–CNR, sempre comprensive e cordiali nel rispondere ad ogni mia richiesta di aiuto. In particolare il Dott. Marco Sacco, il Dott. Luca Greci, il Dott. Gianpaolo Viganò ed il Sig. Francesco Paolucci.

Ringrazio l'ing. Giovanni Pileri dell'azienda Afros s.p.a. per la gentile offerta di un caso applicativo con cui testare il sistema realizzato.

Grazie a tutte le persone che, seppur distanti, mi sono state vicine nel corso degli anni nelle diverse esperienze di formazione e crescita vissute.

Grazie alla mia famiglia per il sostegno e l'incoraggiamento costante durante gli anni dell'università e per avermi sopportato in questi mesi concitati. A loro dedico il traguardo raggiunto.

Sommario

Nel presente lavoro di tesi è stato progettato e realizzato un sistema che si inserisce fra gli strumenti a supporto del personale addetto ad operazioni manuali di manutenzione o di assemblaggio, tipicamente in ambito industriale.
E' stato realizzato un sistema che integra tecniche di Visione Artificiale e di Realtà Aumentata per controllare, guidare e supportare un operatore durante l'espletamento di operazioni manuali. Il sistema, tramite una webcam ed un dispositivo di visualizzazione, valuta in tempo reale la correttezza delle azioni eseguite, visualizza istruzioni, suggerimenti, correzioni ed informazioni visive aggiuntive, avanzando in modo automatico nel flusso logico del processo fino alla completa esecuzione della procedura da svolgere. In particolare è stato necessario sviluppare algoritmi affidabili per identificare ed inseguire oggetti tridimensionali, per riconoscere particolari forme e colori e per analizzare lo stato dell'ambiente di lavoro allo scopo di estrarre informazioni utili, con la particolarità avanzata di rinunciare all'uso di marker precostituiti e di adottare marker naturali per tracciare la posizione della webcam nello spazio e per usufruire correttamente delle funzionalità di Realtà Aumentata che permettono di migliorare l'interazione con l'utente attraverso l'indicazione degli oggetti da usare o delle parti su cui agire grazie all'inserimento di modelli tridimensionali nel video ripreso in tempo reale.
Un'apposita applicazione gestisce la programmazione delle regole di funzionamento del sistema di visione e la determinazione dei parametri delle operazioni di computer vision da eseguire, i dati da analizzare e le azioni da intraprendere.
Una seconda applicazione si occupa, invece, della lettura di tali regole per il controllo del flusso logico del processo nel loro impiego sul campo. Il sistema di visione, opportunamente interrogato, restituisce le informazioni necessarie alla valutazione dell'esecuzione corrente. L'impiego dell'interfaccia grafica in Realtà Aumentata consente la comunicazione con l'utente.
Sono stati infine realizzati alcuni casi applicativi per la validazione delle funzionalità sviluppate e della flessibilità di programmazione del sistema.

Parole chiave: Visione Artificiale, Realtà Aumentata, Natural Features Tracking, Multiple Object Detection and Tracking, Assistenza Manualistica Tecnica, Manutenzione Ordinaria, Assemblaggio Manuale in Serie.

Abstract

The focus of this work is to present the design, implementation and testing of a prototype to support the trained personnel applying maintenance and assembly operations in the industrial domain.
Computer Vision algorithms and Augmented Reality techniques have been integrated to support and control the user conducting manual task. The system developed is made up of a webcam and a display. It is able to automatically verify and adjust the logical flow of the process by controlling the user's actions, to send him feedbacks on the operations done, by showing instructions, suggestions and corrections on the display, till the end of the monitored procedure.
It has been developed a dedicated vision system able to detect and track in real time 3D objects, to recognize shapes and colors, to analyze the scene and find out useful information for the decision support module. This work has also implied the development of a markerless vision-based tracking system which, by detecting natural features inside the scene, allows to retrieve a global spatial reference system for the augmented reality real-time functionalities that can increase the human-computer interaction by highlighting components to handle or by rendering useful text instructions or 3D models.
The system can be easily programmed by a proper application, which manages the operating rules to follow, the data to process and the actions to be done.
Another application is used to read and execute the programmed process flow in real-time during the user's work. The vision system returns all the information needed to assess the user's execution of the current task. The graphic user interface displays the most appropriate feedback in augmented reality.
Then, three test cases have been carried out as validation of the whole system, by focusing on the developed techniques and the programming flexibility.

Keywords: Computer Vision, Augmented Reality, Natural Features Tracking, Multiple Object Detection and Tracking, Service Manual, Routine Maintenance, Manual Assembly.

Indice Generale

Elenco delle figure

Elenco delle Tabelle

Introduzione

La fase di addestramento del personale ha acquisito negli anni sempre più importanza a causa della rapidità con cui si evolvono i mercati e le tecnologie. La crisi ha cambiato il mondo ed i consumi, spingendo sempre di più le aziende a produrre in maniera diversa un maggior numero di prodotti, e, quindi ad una maggiore necessità di formazione del personale tecnico. I mercati sempre più instabili costringono le aziende ad un uso maggiore di personale temporaneo, per il quale è necessaria una formazione efficace, ma al tempo stesso efficiente. Inoltre, in realtà aziendali medio-piccole è molto sentita la necessità di favorire una più agevole e completa trasmissione delle competenze pratiche alla nuova manodopera da parte del personale più esperto.
I sistemi di produzione odierni, grazie alla continua innovazione delle tecnologie e degli impianti, stanno raggiungendo un livello di automazione sempre più alto, e di conseguenza una complessità in continuo aumento, tali da richiedere competenze maggiori per mantenere il sistema efficiente. Personale non all'altezza delle proprie mansioni, infatti, può portare ad un prodotto non competitivo, nonché incidere negativamente sulla qualità dell'output con conseguente ripercussione sulle vendite, e può anche implicare danneggiamenti agli impianti, provocare fermi macchina con il rischio di mettere a repentaglio la sicurezza propria e dei propri colleghi. La formazione del personale è, dunque, un aspetto molto importante all'interno di una realtà aziendale, poiché deve colmare le differenze tra le capacità attuali e le capacità attese per il personale (efficacia), utilizzando la minor quantità possibile di risorse, sia umane che materiali (efficienza).
I metodi classici di addestramento in un'impresa sono: *On the job training*, *Face to face training* e *Computer based training*. Nel primo l'addestrando acquisisce le competenze direttamente sul luogo di lavoro da un operatore esperto. Il secondo consiste in lezioni in aula, seminari e workshop attuati per introdurre nuove tecnologie di lavorazione. Il terzo è un tutorial digitale che attraverso uno scenario interattivo si propone di preparare i tecnici a lavorare su un macchinario reale. Con l'evolversi della tecnologia si è assistito alla nascita di metodi di addestramento innovativi, quali l'*e-learning*, che viene considerato la versione on-line dell'addestramento basato su computer, ed i *metodi basati sull'interazione con sistemi virtuali*, che impiegano tecniche di Realtà Aumentata.
La Realtà Aumentata (Augmented Reality o AR) è una tecnologia che consente di sovrapporre, in tempo reale, oggetti virtuali all'ambiente fisico. È il punto di incontro tra Realtà Virtuale e mondo reale.

L'utente percepisce il mondo con informazioni aggiuntive (testi, immagini 3D) con cui può interagire attraverso semplici dispositivi. La AR, proprio per le sue caratteristiche, si presta all'impiego nell'attività di addestramento.
Grazie al crescente sviluppo di dispositivi mobili di visualizzazione e di elaborazione ed alle tecnologie di Visione Artificiale (o Computer Vision) sono stati sviluppati in ambito militare, aeronautico e auto-motive sistemi in grado di assistere i tecnici nelle attività lavorative. Questi sistemi hanno l'obiettivo di mostrare all'operatore la corretta procedura da seguire e gli strumenti da utilizzare, visualizzando informazioni aggiuntive sui componenti da manipolare o sulle norme di sicurezza da adottare nell'esecuzione delle varie operazioni che compongono i task a lui assegnati.
I principali obiettivi concreti di una tale applicazione sono:

- Incrementare la qualità delle operazioni manuali sul campo;
- Ridurre i costi dei processi di manutenzione e/o assemblaggio (riparazioni più rapide, periodi di inattività abbreviati);
- Ridurre l'impiego di risorse da dedicare all'addestramento del personale;
- Creare una manualistica aggiornata ed efficiente, che permetta di ridurre la necessità di manutenzioni straordinarie;
- Aumentare la sicurezza sul posto di lavoro;
- Implementare la versatilità delle operazioni, permettendo di intervenire su prodotti diversi.

Nonostante l'uso di queste tecnologie sia già stato adottato per aumentare l'efficienza dei processi lavorativi, non è stato rintracciato nello stato dell'arte analizzato un sistema pensato per verificare l'effettivo compimento del passo procedurale assegnato all'operatore.

Il presente lavoro di tesi, realizzato presso l'Istituto di Tecnologie Industriali e l'Automazione del Consiglio Nazionale delle Ricerche (ITIA-CNR), si inserisce all'interno degli strumenti di aiuto e supporto alle attività industriali. Il sistema progettato e realizzato può considerarsi un potenziamento del concetto di sistema manualistico e si propone l'obiettivo di incrementare la qualità dell'interazione uomo-macchina grazie alla visualizzazione su di un display di informazioni sulle operazioni da eseguire ed al rilevamento, in tempo reale, di eventuali errori commessi dall'operatore, avanzando in automatico nel flusso logico della sequenza di operazioni da compiere fino alla completa e corretta esecuzione della procedura da svolgere.
Il presente lavoro di tesi è articolato nelle fasi operative di analisi, progettazione, sviluppo e test ed ha condotto alla realizzazione di:

- Un sistema di "regole di funzionamento" codificate che determinano le operazioni da compiere, i dati da analizzare e le azioni da intraprendere, seguendo una casistica predefinita di condizioni possibili alle quali riferirsi;
- Un'applicazione software per la compilazione, la modifica ed il salvataggio di tale insieme di "regole di funzionamento" che il sistema seguirà per assolvere uno specifico compito (*fase istruttore*);

- Un'applicazione software per il caricamento e l'esecuzione della sequenza, per assistere il personale tecnico addetto al suo svolgimento e controllarne l'esecuzione (*fase operatore*). Il sistema è in grado di valutare la correttezza delle operazioni eseguite dall'operatore ed adattare il flusso logico di conseguenza, mostrando un opportuno feedback all'operatore;
- Una classe di funzioni che gestisca la visualizzazione tramite Realtà Aumentata e le informazioni sul tracciamento per mostrare all'utente in tempo reale le istruzioni su passaggi da effettuare e componenti da manipolare;
- Un sistema di visione in grado di riconoscere features naturali presenti all'interno di un ambiente di lavoro non strutturato (markerless), allo scopo di tracciare la posizione della telecamera nello spazio, attraverso un riagganciamento periodico alle features naturali presenti nella memoria del sistema;
- Un insieme di funzioni ed algoritmi di visione artificiale in grado di identificare ed inseguire oggetti tridimensionali, di riconoscere particolari forme o colori in un'immagine e di analizzare lo stato dell'ambiente di lavoro per estrarre informazioni utili al sistema.

Dato l'ampio spettro di applicazioni si è scelto di sviluppare tutte le funzionalità del sistema in modo da poter essere utilizzate con flessibilità. È possibile, infatti, modificare i parametri di ingresso o l'ordine delle operazioni, oppure la tipologia della condizione da ricercare in modo da programmare il sistema per l'esecuzione in un contesto diverso, con altre esigenze operative.

Il presente lavoro di Tesi si articola in 6 capitoli.
Nella prima parte della Tesi si analizza lo stato dell'arte dei sistemi di Visione Artificiale (capitolo 1), delle tecniche di Realtà Aumentata (capitolo 2) e dei sistemi di assistenza all'operatore (capitolo 3). Dopo una breve introduzione sulle loro potenzialità e finalità, se ne descrivono l'evoluzione e le applicazioni più recenti e si presentano gli algoritmi utilizzati. Nel terzo capitolo si riportano, inoltre, gli esperimenti più rilevanti presenti in letteratura, analizzandone tecnologie utilizzate, obiettivi raggiunti e limiti evidenziati.
Nel capitolo 4 sono presentate le fasi di progettazione e sviluppo del sistema proposto. Se ne descrivono nel dettaglio l'architettura e la logica di funzionamento. Si analizza, inoltre, il processo logico che, dall'individuazione dei requisiti, conduce alla definizione delle varie funzionalità del sistema ed alla loro implementazione, attraverso la parametrizzazione e la codifica dello scambio dati tra macchina ed utente.
L'ultima parte del lavoro consiste nella validazione delle funzionalità sviluppate.
Il capitolo 5 descrive l'impiego del sistema in relazione ad alcuni casi applicativi, per confrontare gli obiettivi proposti con le reali aspettative aziendali. Nel capitolo 6 si presentano i risultati di alcuni test effettuati per validare il sistema di tracciamento sviluppato.

Capitolo 1
La Visione Artificiale

La vista è un senso straordinario e potente perché fondamentale per l'uomo nella percezione spaziale e nell'interazione con l'ambiente circostante.
Da qualche decennio la *Visione Artificiale* sta assumendo sempre più importanza nel mondo industriale e dell'automazione, tanto da permettere lo sviluppo di applicazioni commerciali per l'utenza di massa, grazie al progredire dei risultati ottenuti nelle sue applicazioni e, di pari passo, all'incremento in termini di prestazioni delle tecnologie per l'acquisizione e l'elaborazione delle immagini.
I campi di applicazione della visione artificiale sono molteplici e tutti di grande interesse per la ricerca odierna: video sorveglianza, modellazione di oggetti e ambienti, interazione tra uomo e calcolatore, controllo qualità, etc.
Tra tutti questi utilizzi, quello che ultimamente sta riscontrando una grande attenzione è l'individuazione e l'inseguimento di particolari features ambientali per il controllo di robot e veicoli autonomi, in modo da favorirne la localizzazione e semplificare il difficile task del movimento in un ambiente sconosciuto.

1.1. Definizioni

Non è semplice fornire una completa e soddisfacente definizione di *Computer Vision*, perché il termine stesso racchiude tante e diverse tecnologie e applicazioni.
Si può comunque sintetizzare il concetto nel seguente modo:
Il termine Visione Artificiale riguarda la progettazione di sistemi composti da parti ottiche, elettroniche e meccaniche per esaminare, a partire da immagini [44], sia nello spettro del visibile che non, oggetti o processi industriali con l'obiettivo di prendere decisioni ed effettuare valutazioni, al fine di migliorare la qualità del processo stesso o dei prodotti coinvolti nell'analisi [23].

1.2. Componenti di un sistema di visione

Un sistema di visione è un sistema complesso che integra molteplici componenti di origine diversa.
I principali componenti di un tipico sistema di visione sono[57]:

- *Sistema ottico* e *sistema di illuminazione*. Hanno entrambi il compito, svolto in modo differente, di migliorare le prestazioni del sensore, focalizzando la luce incidente, per la formazione dell'immagine;
- *Sensori* per acquisire l'immagine. Il sensore è una matrice di micro elementi sensibili (pixel) su cui il sistema ottico produce l'immagine (CCD o CMOS) e la trasforma in un segnale elettrico in uscita. Questo segnale viene convertito e salvato sul computer in forma digitale;
- *Sistema di elaborazione dell'immagine*. È costituito da hardware specializzato ed ha il compito di effettuare operazioni primitive, aritmetiche e logiche sull'immagine;
- *Software* dedicato ad *image processing and analysis*, è il cuore di ogni applicazione di computer vision, ovvero è il componente che estrae le informazioni dall'immagine e le elabora al fine di assolvere il compito che l'applicazione si propone;
- *Dispositivo di archiviazione* di massa, che ha il compito di immagazzinare la grande mole di immagini catturate dai dispositivi di visione in attesa di elaborazione;
- *Interfaccia uomo-macchina*. È la componente che permette la comunicazione fra il sistema e l'utente.

1.3. Applicazioni

Un sistema di visione può trovare applicazioni in moltissimi contesti dell'attività umana. Per gli obiettivi del presente lavoro, si affronterà principalmente l'ambito industriale e dell'automazione. Vanno tuttavia ricordate le numerose operazioni nei campi del bio-medicale e della riabilitazione, della sicurezza e dei beni culturali in cui sono coinvolti sistemi di Visione Artificiale. Tra le principali applicazioni in ambito industriale, vi sono:

Controllo di processo
Necessità di tutte le aziende è quella di poter controllare ed automatizzare il flusso di produzione. Per controllo e automatizzazione si intende l'utilizzo di tutti quei sistemi il cui scopo è di guidare un robot industriale nell'afferraggio e nel posizionamento di oggetti da posizione non note a priori, come potrebbe succedere su un nastro trasportatore, oppure robot che devono eseguire lavorazioni su pezzi la cui posizione nello spazio non è determinata a priori. Il principale vantaggio dell'ausilio di sistemi di visione con robot industriale è la capacità di correggere in tempo reale la loro posizione.
Per controllo di processo può anche essere considerato il controllo del traffico, delle infrazioni sulle autostrade, il controllo dei mezzi presenti sulle piste di un aeroporto. Recentemente sistemi di visione sono stati installati in centri commerciali ed in zone pedonali per applicazioni di videosorveglianza oppure per rilevare comportamenti pericolosi all'interno di aree circoscritte.

Misure non a contatto
La Computer Vision è utilizzata anche per compiere misure dimensionali non a contatto per oggetti di natura differente.
L'impiego metrologico della Computer Vision consiste nel riconoscere la forma di un oggetto o nel calcolarne una dimensione caratteristica per valutarne difetti dimensionali. Le numerose applicazioni in questo ambito attestano l'affidabilità, l'accuratezza e la ripetibilità delle misure ottenute da un sistema di visione.

Controllo qualità
Definite le tolleranze del processo, è possibile demandarne il controllo a macchine di visione industriale che permettono il rilevamento sistematico di difetti in modo ripetibile ed accurato, a velocità non confrontabili con il tempo di ispezione umano.

Localizzazione e odometria
Il processo atto a determinare, attraverso l'utilizzo di un sensore visivo, l'orientamento e la posizione di un robot mobile nello spazio 3D a partire da una sequenza di immagini è definito, in letteratura, con l'acronimo SLAM (Simultaneous Localization and Mapping). Naturalmente gli input visivi non sono adatti solo per la navigazione, ma permettono anche lo svolgimento di numerosi altri task. Per esempio, oltre a orientarsi nell'ambiente circostante, un robot potrebbe generare una ricostruzione 3D, movimentare oggetti d'interesse, classificare il terreno, etc.

1.4. Considerazioni preliminari

L'elemento che sta alla base del generico processo di analisi della Computer Vision è un'immagine.
L'elaborazione di un'immagine è un'operazione complessa che necessita di un buon grado di esperienza nel settore.

Alcuni dei problemi che si possono riscontrare nelle applicazioni di Visione Artificiale possono essere:

- perdita di informazioni: il sensore ha una definizione limitata a causa sia del numero di pixel disponibili per la rappresentazione dell'immagine, sia per i limiti tecnologici del trasferimento di informazioni;
- interpretazione: l'uomo riesce ad interpretare senza eccessivo sforzo qualsiasi scena osservi grazie alle proprie capacità di elaborazione cognitiva, di ragionamento di memoria. Un'applicazione di Computer Vision può analizzare ed interpretare solo le immagini per le quali è stata progettata;
- rumore: un sistema di vision, come ogni dispositivo meccanico ed elettronico, è soggetto ad errore, in presenza di disturbi che peggiorano la qualità delle immagini acquisite, quali assenza o eccessiva luce, presenza di occlusioni, disturbi elettromagnetici.

Quelli appena elencati sono solo alcuni dei problemi che si affrontano nello sviluppo di un sistema di computer vision. Altre problematiche verranno affrontate, e risolte, nei prossimi capitoli.

1.5. Calibrazione della telecamera

Per calibrazione di una telecamera si intende quell'operazione che permette di ricavare, determinandoli con un metodo adeguato, tutti i parametri del sistema di visione con il fine di creare un legame tra il piano 2D dell'immagine acquisita dalla telecamera e lo spazio 3D che rappresenta.
Lo scopo ultimo della calibrazione è quello di calcolare la matrice che riassume tutte le informazioni riguardanti la struttura interna della telecamera ed il suo posizionamento nello spazio.
L'idea alla base dei calcoli per ottenere questi parametri consiste nella conoscenza delle proiezioni di punti 3D, di coordinate note (punti di calibrazione), per risolvere le equazioni della proiezione prospettica ottenendo quelli che sono chiamati parametri estrinseci ed intrinseci della telecamera[25].
La conoscenza a priori di un certo numero di corrispondenze non è tuttavia sempre possibile, per questo ci si serve di un oggetto tridimensionale, o *pattern*, di aspetto noto, dalla cui immagine sia possibile estrarre facilmente alcuni punti di interesse per calcolare tali corrispondenze.
Il pattern che solitamente si utilizza durante l'operazione di calibrazione è una scacchiera, dalla quale si andranno a riconoscere gli angoli dei riquadri che la compongono.

1.5.1. Parametri intrinseci

I parametri intrinseci permettono il passaggio da coordinate immagine a coordinate espresse nel sistema di riferimento della telecamera. Per coordinate immagine si intendono i valori, espressi in pixel, che rappresentano la posizione di un punto sul piano immagine [63].
Ogni punto dell'immagine ha un suo corrispondente in un sistema di coordinate solidale con la telecamera. In **Fig. 1.1** è data una rappresentazione schematica dei due sistemi di riferimento.
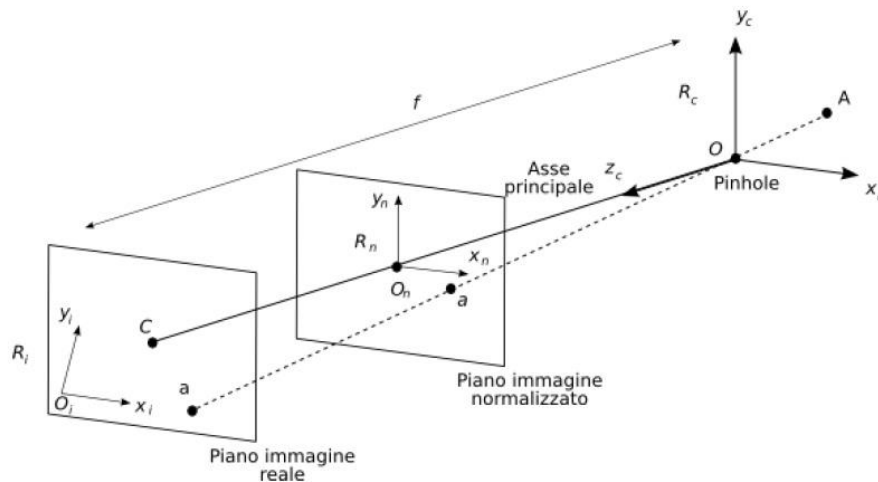
Figura 1.1 – Sistemi di coordinate immagine e telecamera.

I **parametri intrinseci** riassumono le seguenti informazioni:

- lunghezza focale f
- punto principale $C_i(x_C^i, y_C^i)$
- dimensioni dei pixel (s_x, s_y)

Con lunghezza focale si indica la distanza, lungo l'asse principale, tra piano immagine e punto focale. Il punto principale, le cui coordinate sono espresse in pixel nel sistema di riferimento dell'immagine, è il punto in cui l'asse principale incide il piano immagine.
Le dimensioni dei pixel costituiscono dei fattori di scala che tengono conto della forma rettangolare, e non quadrata, dei pixel. Se si pensa ad un piano immagine normalizzato (**Fig. 1.1**) parallelo al piano immagine reale e posto a distanza unitaria dal centro di proiezione, la relazione tra un sistema di riferimento solidale a questo piano, con origine nel punto in cui l'asse principale interseca il piano stesso, e il sistema di riferimento della telecamera, è dato dal sistema di equazioni (1.1) che descrivono la trasformazione prospettica:

$$\begin{cases} x_A^n = \frac{x_A^C}{z_A^C} \\ y_A^n = \frac{y_A^C}{z_A^C} \end{cases} \tag{1.1}$$

dove A è un punto nello spazio. In realtà il piano immagine dista dal *pinhole* (**Fig. 1.1**. e appendice A) della lunghezza focale $f \neq 0$.
Considerando le dimensioni dei pixel s_xe s_y espresse in [pixel/m], l'equazione (1.1) diventa:

$$\begin{cases} x_A^n = s_x \cdot f \cdot \frac{x_A^C}{z_A^C} \\ y_A^n = s_y \cdot f \cdot \frac{y_A^C}{z_A^C} \end{cases} \tag{1.2}$$

Solitamente l'origine del sistema di riferimento immagine non coincide con il punto principale ma, ad esempio, con il vertice in basso a sinistra, come nel sistema di riferimento rappresentato in **Fig. 1.1**.
In questo caso è necessario aggiungere al sistema di equazioni (1.2) un fattore di traslazione dato dalle coordinate del punto principale.

$$\begin{cases} x_A^n = s_x \cdot f \cdot \frac{x_A^C}{z_A^C} + x_C^i \\ y_A^n = s_y \cdot f \cdot \frac{y_A^C}{z_A^C} + y_C^i \end{cases} \tag{1.3}$$

Non sempre gli assi del sistema di riferimento dell'immagine sono esattamente perpendicolari, può esserci tra di loro un angolo tale da rendere i pixel simili a parallelogrammi anziché a rettangoli. Questo termine cambierebbe i fattori di scala del sistema (1.3) e aggiungerebbe un legame tra le due coordinate del sistema di riferimento immagine. Spesso però questo valore viene trascurato perché poco lontano dal valore ideale.
Tutti i parametri intrinseci possono essere riassunti nella matrice **K**, detta *matrice di calibrazione della telecamera*:

$$\begin{bmatrix} s_x \cdot f & 0 & x_C^i \\ 0 & s_y \cdot f & y_C^i \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

Quindi, dato un punto $\tilde{A}_C = [x_A^C \quad y_A^C \quad z_A^C \quad 1]^T$ espresso in coordinate omogenee nel sistema di riferimento delle telecamere, il corrispondente punto $\tilde{A}_i = [x_A^i \quad y_A^i \quad 1]^T$ in coordinate immagine è dato dall'equazione (1.5).

$$\tilde{A}_i = [K \quad 0] \cdot \tilde{A}_C \tag{1.5}$$

1.5.2. Parametri estrinseci

I parametri estrinseci mettono in relazione il sistema di coordinate solidale con la telecamera con il sistema di riferimento mondo (**Fig. 1.2**).
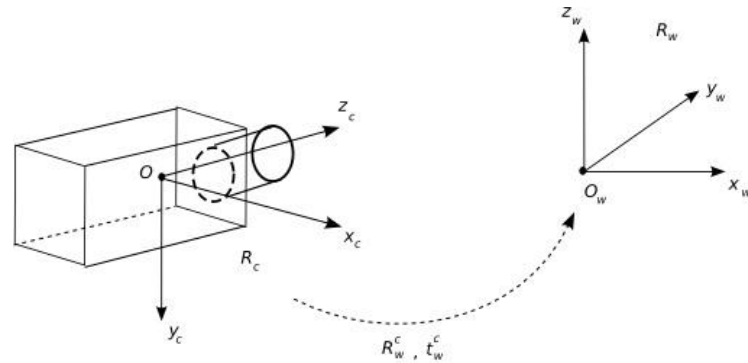
Figura 1.2 – Parametri estrinseci.

Essi sono:

- matrice di rotazione R_w^c
- vettore di traslazione t_w^c

cioè le trasformazioni che portano il sistema di riferimento telecamera a coincidere con il sistema di riferimento mondo. I parametri estrinseci sono riassumibili in una matrice di rototraslazione E:

$$E = \begin{bmatrix} R_w^c & t_w^c \\ 0 & 1 \end{bmatrix} \tag{1.6}$$

Dato il punto A nello spazio e la sua rappresentazione in coordinate omogenee nel sistema di riferimento mondo $\tilde{x}_A^w = [x_A^w \quad y_A^w \quad z_A^w \quad 1]^T$, il corrispondente punto $\tilde{x}_A^i = [x_A^i \quad y_A^i \quad 1]^T$ in coordinate immagine è dato dall'equazione (1.7), a meno di un fattore di scala dovuto al fatto che tutti i punti dello spazio appartenenti allo stesso raggio luminoso corrispondono a un unico punto dell'immagine.

$$\tilde{x}_A^i = P \cdot \tilde{x}_A^w \tag{1.7}$$

Con **P** si indica la matrice di trasformazione proiettiva.

$$P = K \cdot [R \mid t] \tag{1.8}$$

1.5.3. Calcolo dei parametri di calibrazione

L'idea che sta alla base della maggior parte dei metodi di calibrazione consiste nel ricavare i parametri della telecamera risolvendo un sistema di equazioni che lega un insieme di coordinate di punti 3D alle loro proiezioni sull'immagine. Questo significa calcolare la matrice di trasformazione proiettiva e la sua decomposizione in parametri intrinseci ed estrinseci.

Per avere l'insieme di punti 3D si utilizza un pattern di calibrazione con geometria nota che viene fotografato più volte, in diverse posizioni e orientazioni, dalla telecamera, per ottenere i corrispondenti punti 2D. In alcuni casi è utile che siano note anche le posizioni del pattern nello spazio[65].

Il processo per il calcolo dei parametri di calibrazione si divide in due parti fondamentali: il calcolo della matrice di trasformazione prospettica e la stima dei parametri intriseci ed estrinseci a partire da questa matrice. La prima fase consiste nello stabilire, tra punti nello spazio e punti dell'immagine, la seguente relazione:

$$\begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \quad (1.9)$$

Gli elementi della matrice possono essere determinati a partire dal legame tra le coppie di punti attraverso i parametri di calibrazione. Questa matrice è anche detta *omografia*.

Una volta costruita l'omografia è necessario risolvere un sistema di equazioni in cui le incognite sono gli elementi della matrice, da cui verranno poi ricavati i parametri. Per fare questo è possibile utilizzare le tecniche ai minimi quadrati, che hanno lo scopo di minimizzare lo scarto quadratico medio tra features dell'immagine osservate e predette.

La letteratura propone numerose tecniche per stimare la matrice di proiezione prospettica, classificabili in due categorie principali (**Fig. 1.3**):

- metodi che utilizzano una immagine di molti (almeno due) piani contenenti un pattern noto;
- metodi che utilizzano molte (almeno tre) immagini diverse di uno stesso pattern piano.

Da un punto di vista pratico, la scelta di un pattern planare, quale una scacchiera, risulta più semplice, rispetto all'individuazione di un oggetto tridimensionale adatto, con piani perfettamente ortogonali su cui dovrebbe essere posto il pattern.

Per quanto detto, si è scelto di utilizzare la tecnica di calibrazione proposta da Zhang (**Fig. 1.4**) ed implementata nella maggior parte delle librerie dedicate alla computer vision, proprio perché risulta un buon compromesso alle richieste di semplicità realizzativa e precisione[72].
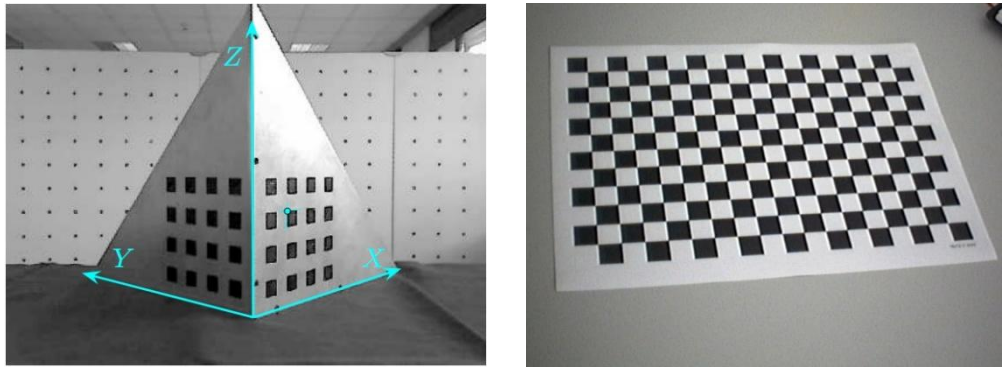
Figura 1.3 – Esempi di pattern di calibrazione.
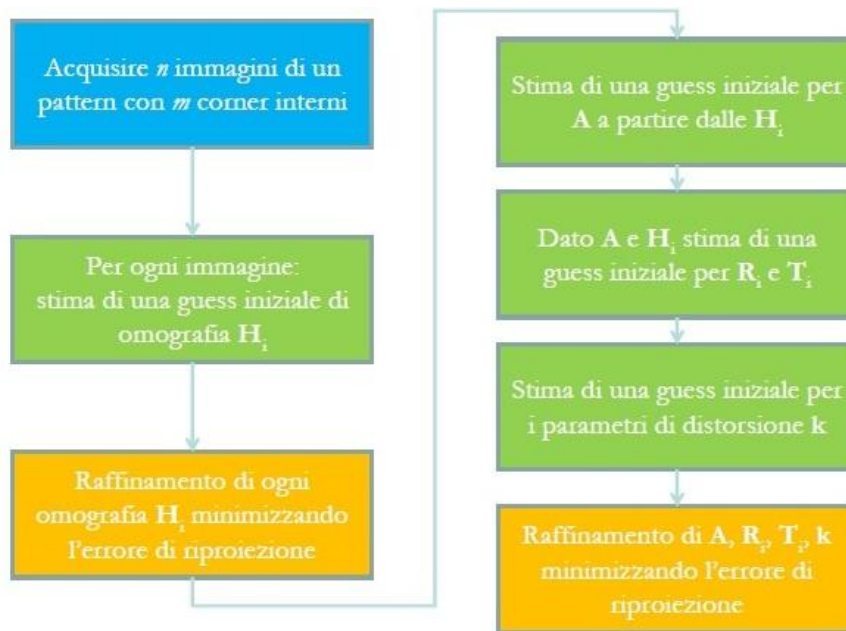
Figura 1.4 – Algoritmo del metodo di calibrazione di Zhang.

1.5.4. Metodo di Zhang

La tecnica di calibrazione proposta da Zhang[73] utilizza un approccio a metà tra le due categorie di metodi descritte nei paragrafi precedenti perché lavora con posizioni note di punti 2D e non di punti 3D o punti di cui non si hanno informazioni esplicite. Questo metodo richiede la conoscenza a priori del numero di corner interni della scacchiera (lungo le due dimensioni) e la lunghezza del lato dei quadrati che la compongono.
Su questo metodo si basano le funzioni per la calibrazione implementate nel sistema sviluppato in questa tesi, di cui si parlerà in maniera approfondita nel seguito, ed in

numerosi programmi commerciali di calibrazione, quale il *Calibration Toolbox* di *Matlab*[91].
Come espresso nell'equazione (1.7), le coordinate omogenee 2D dell'immagine $\tilde{x}_A^i$ sono legate alle coordinate 3D mondo $\tilde{x}_A^w$ dalla matrice di trasformazione prospettica.
Si assume che il piano in cui viene posto il pattern di calibrazione per il calcolo dei parametri sia il piano z=0 del sistema di coordinate mondo. Con questa assunzione il legame tra le coordinate si riduce all'equazione (1.10).

$$\begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix} = \boldsymbol{K}[r_1 \quad r_2 \quad r_3 \quad t] \begin{bmatrix} x^w \\ y^w \\ 0 \\ 1 \end{bmatrix} = \boldsymbol{K}[r_1 \quad r_2 \quad t] \begin{bmatrix} x^w \\ y^w \\ 1 \end{bmatrix} \qquad (1.10)$$

Dove $\boldsymbol{H} = \boldsymbol{K}[r_1 \quad r_2 \quad t]$ è la matrice di omografia ed è definita a meno di un fattore di scala. La matrice di rotazione nell'espressione è rappresentata attraverso le sue colonne.
La matrice omografia ha 8 gradi di libertà e contiene l'informazione su 6 parametri estrinseci (3 di traslazione e 3 di rotazione). Sull'omografia è possibile individuare 2 vincoli dati dalla caratteristica di ortonormalità tra le colonne della matrice di rotazione.
Non tutti i metodi dispongono di equazioni e vincoli sufficienti per determinare subito tutti i parametri di calibrazione. Quindi è possibile assumere le coordinate del punto principale, ad esempio come coincidenti con il centro dell'immagine, e trovare le giuste coordinate in un secondo momento.
La soluzione del sistema che si viene a creare prevede l'utilizzo di una soluzione analitica seguita da una tecnica di ottimizzazione non lineare basata sul criterio di massima verosimiglianza.
La soluzione analitica inizia con la costruzione della matrice **B**:

$$\boldsymbol{B} = \boldsymbol{K}^T \cdot \boldsymbol{K}^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{F_x^2} & 0 & -\frac{C_x}{F_x^2} \\ 0 & \frac{1}{F_y^2} & -\frac{C_y}{F_y^2} \\ -\frac{C_x}{F_x^2} & -\frac{C_y}{F_y^2} & \frac{C_x}{F_x^2} + \frac{C_y}{F_y^2} + 1 \end{bmatrix} \qquad (1.11)$$

in cui $F_x = s_x \cdot f$ e $F_y = s_y \cdot f$ rappresentano i coefficienti che tengono conto delle dimensioni dei pixel e della lunghezza focale. C_X e C_y rappresentano le coordinate del punto principale espresse nel sistema di riferimento dell'immagine.
Poiché simmetrica, **B** è caratterizzata da 6 elementi distinti che si possono riassumere nel vettore:

$$\boldsymbol{b} = [B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}] \qquad (1.12)$$

Indicando con h_i la i-esima riga della matrice **H**, si ha:

$$h_i^T \cdot \boldsymbol{B} \cdot h_j = v_{ij}^T \cdot \boldsymbol{b} \tag{1.13}$$

Utilizzando i vincoli sull'*ortonormalità* delle colonne della matrice di rotazione, si ottiene un sistema di equazioni omogenee di dimensione *2n*, dove *n* è il numero di immagini del pattern di calibrazione utilizzate. Si ottiene una soluzione unica di **b** se e solo se $n \geq 3$. Una volta stimato **b** si possono calcolare i parametri intrinseci ricordando la relazione 1.11 valida a meno di un fattore di scala λ[12].

$$C_y = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - {B_{12}}^2} \tag{1.14}$$

$$\lambda = B_{33} - \frac{{B_{13}}^2 + C_x(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \tag{1.15}$$

$$F_x = \sqrt{\frac{\lambda}{B_{11}}} \tag{1.16}$$

$$F_y = \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - {B_{12}}^2}} \tag{1.17}$$

$$C_x = -\frac{B_{13}F_x^2}{\lambda} \tag{1.18}$$

Una volta nota la matrice dei parametri intrinseci **K** è possibile passare al calcolo dei parametri estrinseci:

$$r_1 = \lambda\, K^{-1} h_1 \tag{1.19}$$

$$r_2 = \lambda\, K^{-1} h_2 \tag{1.20}$$

$$r_3 = r_1 \times r_2 \tag{1.21}$$

$$t = \lambda\, K^{-1} h_3 \tag{1.22}$$

1.5.5. Effetti della distorsione delle lenti nell'immagine

La distorsione è un effetto dovuto alla presenza, sulla telecamera, di un obiettivo, cioè un sistema di lenti. Esistono due tipi di distorsione: radiale e tangenziale. L'effetto radiale è più evidente, soprattutto allontanandosi dal centro dell'immagine, dove le linee rette tendono a diventare curve. L'effetto della distorsione sulle immagini è visibile in **Fig. 1.5**.

Le distorsioni si modellano matematicamente con serie infinite i cui termini contengono dei pesi che sono i coefficienti di distorsione. Queste serie sono funzioni che legano la posizione di un punto in un'immagine reale distorta e una ideale non distorta. Spesso è sufficiente fermarsi ai primi due termini della serie per modellare adeguatamente il fenomeno, considerando più termini, però, esiste anche il rischio di instabilità numerica.

Il legame tra un punto reale (X,Y) e ideale (x,y), dal punto di vista della distorsione è dato dalle equazioni:

$$X = x + x[k_1 r^2 + k_2 r^4] + [2p_1 xy + p_2(r^2 + 2x^2)] \quad (1.23)$$

$$Y = y + y[k_1 r^2 + k_2 r^4] + [2p_1 xy + p_2(r^2 + 2x^2)] \quad (1.24)$$

dove $r^2 = x^2 + y^2$.

Anche i coefficienti di distorsione vengono stimati dal metodo di Zhang, descritto nel parametro precedente. All'interno del programma di visione essi verranno stimati in modo da poter ridurre l'effetto della distorsione nelle immagini acquisite.
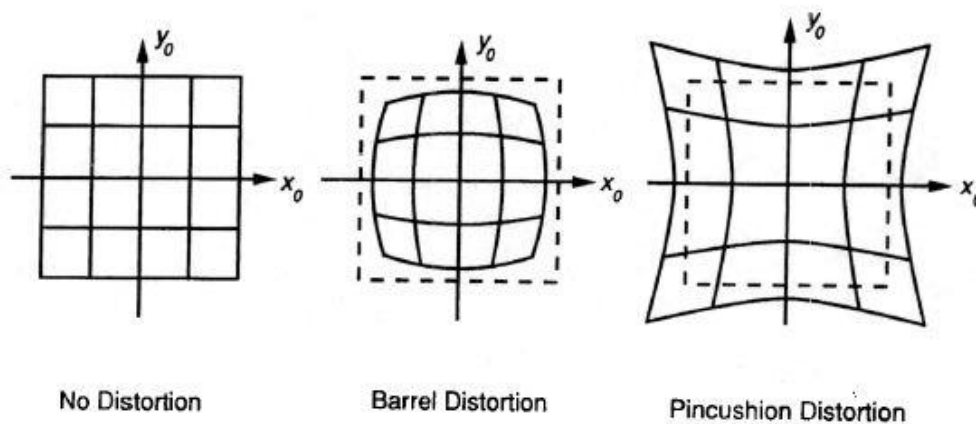
Figura 1.5 – Effetti della distorsione delle lenti.

1.6. Object detection and tracking

Il riconoscimento di oggetti che si muovono nello spazio all'interno di un flusso video è uno dei più importanti e affascinanti compiti della visione artificiale, nonché un argomento di ricerca in aggiornamento continuo.
Nella sua forma più semplice, il tracking può essere definito come un duplice problema: da un lato, la stima della traiettoria di un oggetto nel piano immagine mentre si sposta lungo la scena, dall'altro, l'estrazione di informazioni sulla posizione e l'orientamento dell'oggetto rispetto al sistema di riferimento tridimensionale della telecamera.
Qualunque sia l'obiettivo ultimo dell'analisi, le operazioni che un sistema di tracking deve eseguire sono le seguenti: identificare un oggetto (detection) ed inseguirlo, frame dopo frame, nel suo moto (tracking), analizzare il moto dell'oggetto per riconoscerne il comportamento.
Il tracciamento di oggetti, perciò, si rivela strategico in numerose applicazioni, quali la video-sorveglianza automatizzata, il riconoscimento di gesti e volti, il monitoraggio del traffico, la guida autonoma di veicoli e robot.
Diversi approcci sono stati seguiti per sviluppare sistemi di tracking, ognuno dei quali presenta caratteristiche positive o negative in funzione della tecnologia impiegata. A volte una buona strategia consiste nell'integrare sistemi diversi affinché i limiti di un sistema vengano compensati dalle qualità dell'altro e viceversa. Pertanto, non è possibile affermare a priori quale sia la strategia preferenziale per il calcolo dei sei gradi di libertà che definiscono la posizione e l'orientamento della telecamera rispetto alla scena.
Allo stato dell'arte esistono due principali classi di tecniche con cui, fino ad oggi, si è cercato di affrontare il tracking basato sulla visone artificiale: i metodi *dense motion based* e quelli *feature based*[36].
Gli algoritmi basati sul *dense motion*, anche conosciuto come *optical flow*[10], cercano di stimare il movimento analizzando i vettori di flusso ottico all'interno di un video, ovvero i vettori che rappresentano lo spostamento di alcuni punti particolari dell'immagine (flusso ottico sparso) o di tutti i pixel dell'immagine (flusso ottico denso). I campi di flusso calcolati da questi metodi sono tipicamente utili per applicazioni di basso livello, come l'*obstacle avoidance* (identificazione e aggiramento di ostacoli) poiché risulta molto complicato correlarli con la geometria globale della scena inquadrata.
I metodi *feature based*, invece, tengono traccia solo di un piccolo numero di punti, che corrispondono ad elementi di interesse della scena, detti features, tra due immagini successive del flusso video.
Le features possono essere:

- una proprietà globale dell'immagine o parte di essa, per esempio un livello medio di grigio o un'area in pixel (***global features***);
- una parte dell'immagine con delle proprietà distintive (***local features***).

Come si può facilmente immaginare, l'uso delle features permette una miglior efficienza di calcolo e riduce la quantità di dati prelevati dai frame di cui ci si deve occupare, rendendo così più realistica la possibilità di ottenere performance in tempo reale.
I metodi basati sulle features si appoggiano principalmente sul rilievo della posizione dei punti d'interesse anziché sul confronto diretto dei dati delle immagini, risultando più robusti alle eventuali variazioni di illuminazione della scena inquadrata e al rumore dovuto all'acquisizione. Inoltre essi consentono uno spostamento più ampio tra due immagini successive.
Di seguito si illustrerà la tecnica *feature based* applicata alle immagini acquisite dal sistema e verranno poi analizzati i passi fondamentali nei loro caratteri generali, essendo sempre gli stessi in qualunque algoritmo sviluppato e differenziandosi soltanto nel modo in cui vengono implementati. In ogni sistema di tale tipo analizzato dalla letteratura esistente, infatti, si sono evidenziate quattro fasi costanti (**Fig. 1.6**):

1. **Estrazione delle features**: i punti d'interesse dell'immagine acquisita vengono selezionati e ad ognuno si associano informazioni caratteristiche;

2. **Matching**: ricerca delle corrispondenza tra i punti estratti in precedenza e quelli estratti da un' immagine di riferimento;

3. **Tracking**: procedimento che tiene traccia nelle acquisizioni successive delle features riconosciute nell'immagine di partenza;

4. **Stima del movimento**: algoritmo che, dati due set di coordinate 3D riferite alle posizioni delle features rispettivamente prima e dopo lo spostamento, stima il movimento avvenuto tra le due immagini o la trasformazione prospettica che lega l'immagine corrente a quella di riferimento.

Figura 1.6 – Fasi di un generico algoritmo feature-based.

Nel seguito di questo capitolo, quindi, si tenterà di chiarire il significato dei vari passi del processo e di fare una panoramica delle tecniche e degli algoritmi utilizzati in letteratura per sviluppare tali procedure.

1.6.1. Estrazione di features

La stima del movimento in una sequenza di immagini in molti casi risulta essere un problema mal posto. Se si considerano piccole regioni di un'immagine o anche singoli pixel, solitamente l'informazione disponibile non è sufficiente per determinare lo spostamento in maniera affidabile[61]. Considerando, per esempio, una piccola parte di un'immagine con una zona di bordo lineare e continua (**Fig. 1.7**) è molto difficile trovare la posizione corrispondente in un'immagine più grande, perché il *pattern* si ripete lungo tutta la linea di bordo.
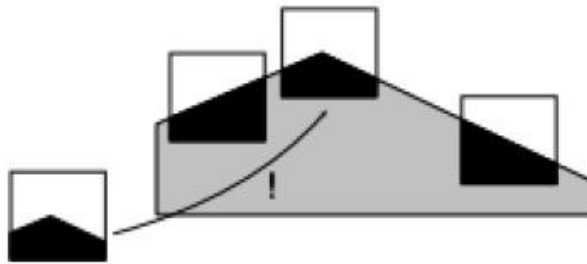
Figura 1.7 – Problema dell'apertura.

In questo caso, infatti, l'*edge* è caratterizzato da una sola direzione, perpendicolare alla linea di bordo, lungo la quale l'immagine è uniforme, rendendo impossibile la ricerca del miglior punto di matching. Questo non determinismo della soluzione è noto in letteratura come *"problema dell'apertura"*[16].
La possibilità di determinare in modo univoco e affidabile la posizione del pattern aumenta considerevolmente se quest'ultimo mostra variazioni in due direzioni differenti. In **Fig. 1.8**, per esempio, la posizione della feature può essere determinata inequivocabilmente in entrambe le direzioni.
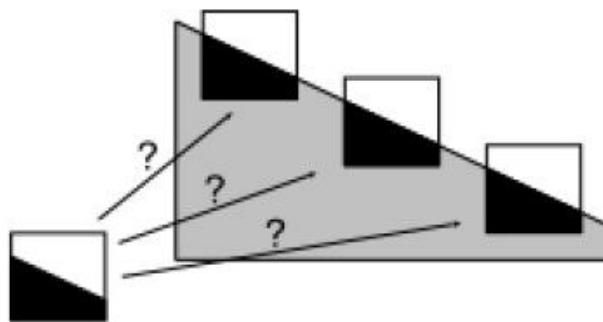
Figura 1.8 – Pattern non univocamente identificabile.

Da ciò si deduce che l'idea alla base dei metodi *feature based* è quella di concentrarsi solo su un piccolo insieme di *interest points*, cioè sui punti che introducono poca incertezza nel calcolo della soluzione.

Figura 1.9 – Porzioni d'immagine adatte per la collocazione di punti d'interesse.

I punti d'interesse utilizzati per gli algoritmi di visione dovrebbero essere collocati in posizioni in cui il contenuto dell'immagine attorno alla feature rilevata permetta di identificare la posizione corrispondente in una seconda immagine con una buona affidabilità. Di conseguenza, le "*buone features*" devono necessariamente mostrare un cambiamento di contenuto immagine in almeno due direzioni, in modo che possano essere localizzate in maniera precisa. Nella pratica, corner, giunzioni a "T", texture complesse forniscono una buona localizzazione (**Fig. 1.9**).
Tuttavia, i punti d'interesse vengono selezionati solo localmente e, pertanto, un punto che a prima vista sembrerebbe semplice da seguire nella fase di tracking potrebbe rivelarsi una cattiva scelta se fossero presenti ambiguità nell'immagine, per esempio, a causa di pattern ripetitivi (**Fig. 1.10**).
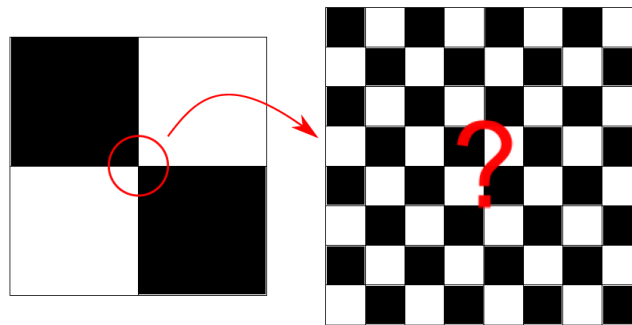
Figura 1.10 – Difficoltà di localizzazione di un corner ripetitivo.

Scegliere le features giuste, pertanto, gioca un ruolo fondamentale nel tracking.
In generale, le migliori features sono quelle che caratterizzano in modo distintivo un oggetto all'interno dell'immagine. La selezione delle features è strettamente correlata alla scelta del tipo di rappresentazione dell'oggetto che verrà utilizzato.
In linea di massima, le *image features* dovrebbero essere locali, significative e rilevabili.

- ✓ **Significative**, perché devono essere associate ad elementi interessanti della scena attraverso il processo di formazione dell'immagine. Tipici esempi di caratteristiche significative sono le forti variazioni di intensità legate al contorno degli oggetti presenti nella scena.

- ✓ **Rilevabili**, perché localizzabili tramite algoritmi esistenti. Features differenti sono associate a differenti algoritmi di rilevamento, questi algoritmi restituiscono come output un descrittore della feature, che ne specifica la posizione ed altre caratteristiche essenziali all'interno dell'immagine.

Fino ad oggi sono stati sviluppati numerosi algoritmi per la rilevazione di features che sostanzialmente si differenziano per la replicabilità delle features estratte dall'immagine e per il carico computazionale. Alcune features possiedono anche le caratteristiche di essere invarianti alle rotazioni e/o ai cambiamenti di scala.
Le tecniche adottate per individuare ed estrarre gli elementi d'interesse sono sostanzialmente due e si distinguono per l'impiego o meno di particolari geometrie fiduciali facilmente rilevabili, detti *marker*.

1.6.1.1. Il tracciamento marker-based

L'utilizzo di geometrie fiduciali facilmente rilevabili nella fase di elaborazione dell'immagine facilita il funzionamento e l'implementazione dell'applicazione. Solitamente si suddividono le geometrie fiduciali in due categorie :

- punti fiduciali;
- piani fiduciali;

In genere i punti fiduciali sono piccole sfere ricoperte di materiale riflettente che ne facilita l'individuazione.
I marker planari sono invece forme quadrate o rettangolari, di solito messe a contrasto di colore (nero su bianco) con la zona che le circonda[35]. Gli spigoli sono individuati all'intersezione di linee ottenute per interpolazione durante la fase di elaborazione. Da ogni spigolo si ricava la corrispondenza tra immagini successive e la posizione viene stimata con l'impiego dei filtri di Kalman, algoritmi per la stima e la misura del moto real-time. I riferimenti bibliografici [31] e [32] dimostrano come un singolo marker di tipo planare sia sufficiente a stimare la sua posizione rispetto alla telecamera. Questo tipo di approccio ha avuto successo grazie ai costi contenuti, alla velocità con la quale si riesce ad avere un sistema di tracking tridimensionale ed, infine, grazie all'implementazione di questo metodo in librerie software gratuite, come ad esempio le librerie *ARtoolkit*[84].
La tecnica marker-based è largamente impiegata in applicazioni di Realtà Virtuale e presenta come vantaggio principale la possibilità di ottenere in tempo reale la posizione e l'orientamento dell'utente. Per contro, presenta i seguenti svantaggi:

- Lo spostamento dei sensori dalla loro posizione originale provoca situazioni di incertezza nei risultati;
- Particolare difficoltà nel posizionare tali dispositivi in alcuni ambienti, come ad esempio, su una macchina utensile;
- Rigidità nei movimenti dell'utente derivanti dal condizionamento fisico e psicologico legato alla presenza di tali sensori;

Da questa breve analisi si capisce che, per gli scopi proposti nel presente lavoro, l'uso di un sistema che impieghi dei marcatori per localizzare e tracciare la posizione dell'utente che si muove in una scena non sia conveniente.
L'intero sistema proposto in questa tesi è basato su di una strategia più flessibile, che non fa uso di marker fiduciali. E' pertanto possibile un suo utilizzo in numerosi e diversi contesti, senza particolari vincoli da porre all'utente prima di operare.
L'idea alla base del sistema proposto è quella di sostituire il marker fiduciale con più features naturali fisse nella scena, per l'identificazione di un sistema di riferimento globale ed assoluto, con cui operare.

1.6.1.2. Il tracciamento markerless

In letteratura sono disponibili numerosi estrattori di features sufficientemente affidabili per effettuare il tracking di alcuni particolari, presenti nella scena, da utilizzare come marker naturali e permettere così il calcolo della posizione della telecamera che li inquadra.
Le proprietà estremamente importanti per un buon feature-detector sono due.
La prima è la **ripetibilità**, cioè la capacità dell'estrattore, in immagini che mostrano lo stesso contenuto da un punto di vista poco differente, di rilevare le stesse features in entrambi i frame.
Un secondo importante criterio per stabilire la bontà di un estrattore è l'**invarianza** dei punti rilevati alle trasformazioni (di scala e rotazionali) presenti tra le due immagini.
Questi fattori risultano poco rilevanti quando le acquisizioni dei frame sono molto ravvicinate nel tempo, in quanto le trasformazioni tra le immagini sono di piccola entità, ma si dimostrano invece determinanti per il tracking delle features quando intercorre un lasso di tempo significativo fra due frame successivi.
Lo studio di descrittori efficienti è un'area di ricerca attiva. La scelta degli algoritmi dipende dal tipo di applicazione che si intende sviluppare. La letteratura offre un'ampia gamma di analisi[66] su base comparativa che ne valutano le prestazioni e
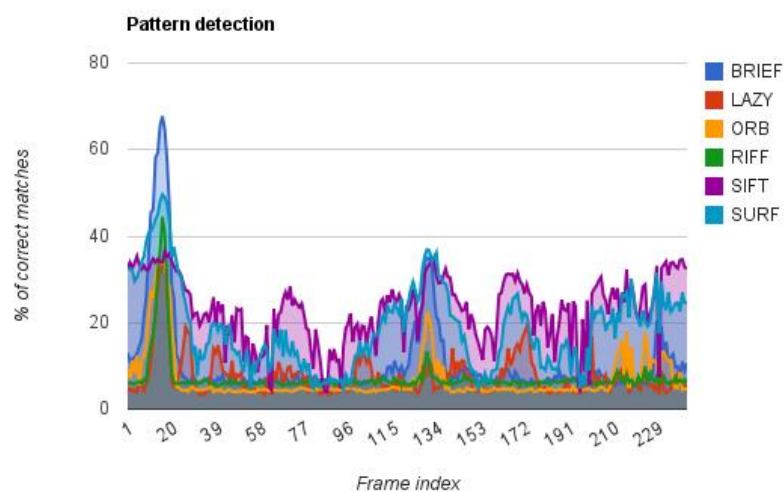
Figura 1.11 – Confronto fra descrittori per l'object detection.

che sono state oggetto di considerazioni preliminari rispetto al percorso intrapreso.
Le immagini seguenti mostrano un confronto diretto fra alcuni algoritmi presenti in letteratura ([9][13][41][54]) nell'individuare un pattern noto, all'interno di un flusso video: in **Fig. 1.11** si valuta l'accuratezza del riconoscimento, in **Fig. 1.12** il costo computazionale degli algoritmi. Si osserva che l'algoritmo che presenta il miglior compromesso tra costi e performance è il SURF, di cui si parlerà nel paragrafo successivo.
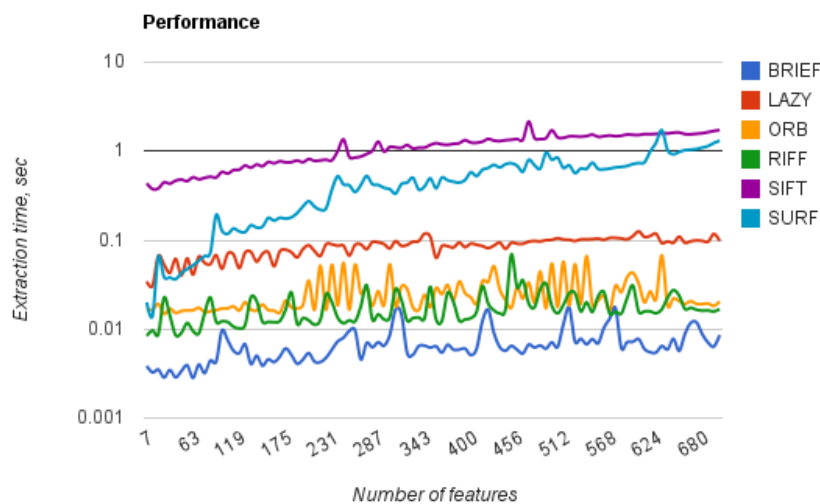
Figura 1.12 – Confronto sul costo prestazionale degli algoritmi.

1.6.1.2.1. Speeded-Up Robust Features (SURF)

L'estrattore SURF[9] è uno dei più recenti algoritmi di estrazione di features nelle immagini. Nato come evoluzione dell'algoritmo SIFT (Scale Invariant Feature Transform) di David Lowe[41], il SURF associa ai punti d'interesse individuati nel frame un descrittore, cioè un vettore caratteristico dell'intorno di tali pixel, che permette di riconoscere una particolare feature dalle altre, in modo robusto rispetto al rumore ambientale, agli errori di rilevazione e alle deformazioni geometriche che avvengono tra le immagini. Il SURF sfrutta una rappresentazione intermedia dell'immagine da processare, detta "*integral image*", calcolata a partire dal frame originale per velocizzare in maniera cospicua il calcolo delle finestre necessarie all'estrattore[39].
La costruzione dei descrittori avviene in due fasi: per ogni punto d'interesse viene individuata l'orientazione principale e, in seguito, si costruisce una finestra centrata sul punto stesso, dalla quale, tramite l'uso congiunto dei *Filtri di Haar* e delle immagini integrali, si estrae un vettore di 64 componenti, riducendo notevolmente sia il tempo per il calcolo che quello per il matching delle features[40].
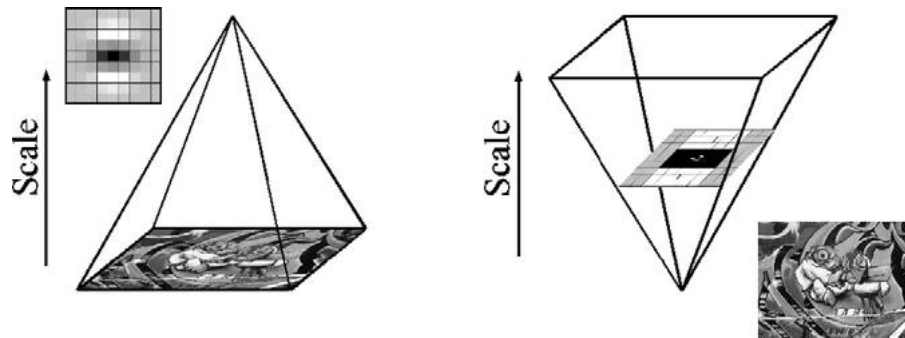
Figura 1.13 – Analisi dello spazio dei fattori di scala in differenti versioni.

I descrittori calcolati vengono poi utilizzati per trovare le corrispondenze tra le features di immagini diverse, confrontando particolari caratteristiche, quali, ad esempio, la distanza euclidea che esiste tra questi vettori.
La caratteristica di associare al punto un vettore descrittivo rende il SURF molto pesante dal punto di vista computazionale, ma gli conferisce ripetibilità e robustezza maggiori rispetto ad altri algoritmi, nonché una buona invarianza alle rotazioni, ai cambiamenti di scala e ai cambi d'illuminazione nell'immagine[22].
Pur basando la sua informazione sulla distribuzione spaziale del gradiente, analogamente al SIFT, questo estrattore risulta migliore nella maggior parte dei casi. La sua superiorità è dovuta al fatto che il SURF integra l'informazione sul gradiente calcolato su sottofinestre, mentre il SIFT dipende solo dalle orientazioni dei gradienti individuali, risultando più sensibile al rumore.
Inoltre, grazie alla continua evoluzione delle capacità computazionali dell'hardware, anche detector come il SIFT e il SURF possono essere utilizzati senza dover rinunciare ad applicazioni real-time o quasi real-time.
Per una trattazione approfondita dell’algoritmo, si rimanda alla bibliografia originale[9].
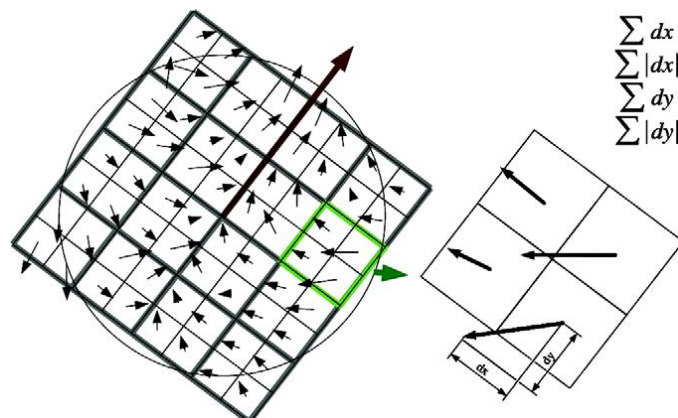
Figura 1.14 – Componenti del descrittore calcolati per ogni finestra.

1.6.1.2.2. Dominant Orientation Templates (DOT)

Questo algoritmo si basa su un semplice approccio di tipo template matching: noto il *template* dell'oggetto, l'algoritmo lo cerca all'interno di un'immagine, anche molto complessa, per determinarne la presenza e la posizione.
L'idea alla base del calcolo è quella di misurare la similarità tra un'immagine di input *I*, e un'immagine campione *O* di un oggetto posizionato all'interno dell'immagine *I*, confrontando l'orientazione dei loro gradienti, come mostrato in **Fig. 1.15**.
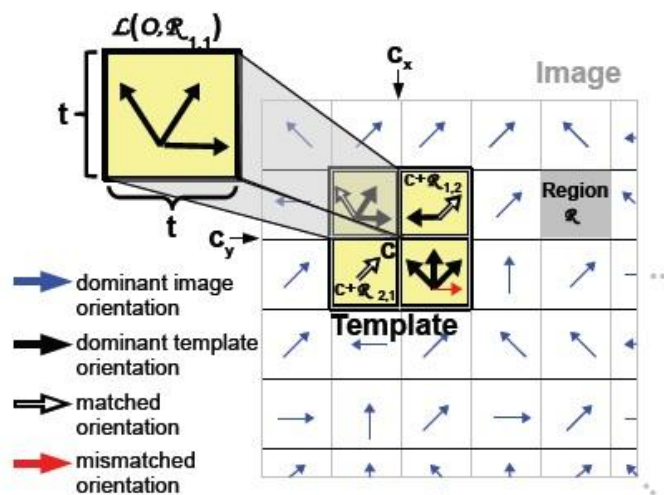
Figura 1.15 – Misura della corrispondenza tra gradienti.

L'algoritmo DOT è stato sviluppato dal dipartimento di *Computer Science* dell'Università TUM di Monaco di Baviera.
Gli sviluppatori [28] hanno scelto di considerare i gradienti delle immagini perché, oltre a rappresentare una descrizione univoca del template, risultano meno sensibili ai cambiamenti di illuminazione ed al rumore. Per rendere ulteriormente robusto l'algoritmo viene utilizzato il loro modulo per considerare la direzione dei gradienti più forti senza che il loro valore venga utilizzato per farne il matching. Inoltre, per gestire correttamente i punti estremi di un oggetto, vengono considerate solo le orientazioni dei gradienti e non le loro direzioni. In questo modo la misura non è condizionata dallo sfondo su cui è posto l'oggetto (sfondo di colore uniforme chiaro o scuro). Come mostrato in **Fig. 1.16**, il template è fatto scorrere sopra l'immagine per calcolare la risposta ad ogni posizione dell'immagine. Se questa risposta è superiore ad una certa soglia, viene considerata positiva la ricerca del template.
In **Fig. 1.17** si può osservare il funzionamento on-line dell'algoritmo.
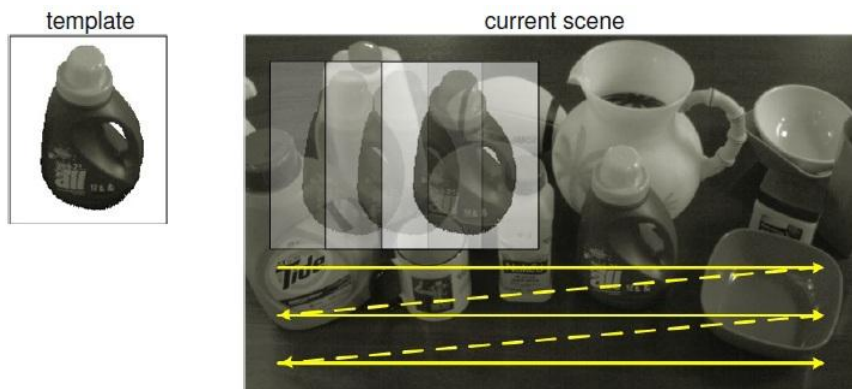
Figura 1.16 – DOT: ricerca del template all'interno dell'immagine.

Rispetto ad altri metodi di template matching, spesso lenti per applicazioni real-time, DOT aumenta la velocità di matching grazie all'impiego di tre tecniche:

- **discretizzazione dell'orientazione del gradiente**

Le informazioni del gradiente per ogni pixel vengo discretizzate in modo da essere contenute in un byte: 7 bit sono utilizzati per l'orientazione, 1 bit per il modulo (sotto/sopra soglia). Questo processo però è differente tra template e immagine: nell'immagine, ad ogni pixel è associato il valore del gradiente discretizzato, mentre per il template, ad ogni pixel viene associato il valore del maggior gradiente discretizzato di un'area di dimensione 7x7 pixel.

- **sotto-campionamento dell'immagine**

Sia il template che l'immagine vengono suddivisi in regioni di dimensione 7x7 pixel. Per ogni pixel della regione viene calcolato un byte (informazione del gradiente). Tutte queste informazioni verranno combinate assieme tramite un *OR* logico per descrivere la regione intera.

- **ottimizzazione tramite istruzioni SSE**

Le Streaming Simd Extension sono un'insieme di istruzioni che danno al calcolatore la possibilità di calcolo parallelo per operazioni in virgola mobile, in modo da poter gestire operazioni in cui con una singola istruzione sia possibile processare più dati, non più elaborati in modo sequenziale. Le SSE risultano utili nella manipolazione di contenuti multimediali, nella decompressione dei filmati e nell'elaborazione di geometrie 3D. Il loro impiego nell'algoritmo serve per velocizzare il tempo computazionale della comparazione dei byte array (realizzata tramite operatori logici).
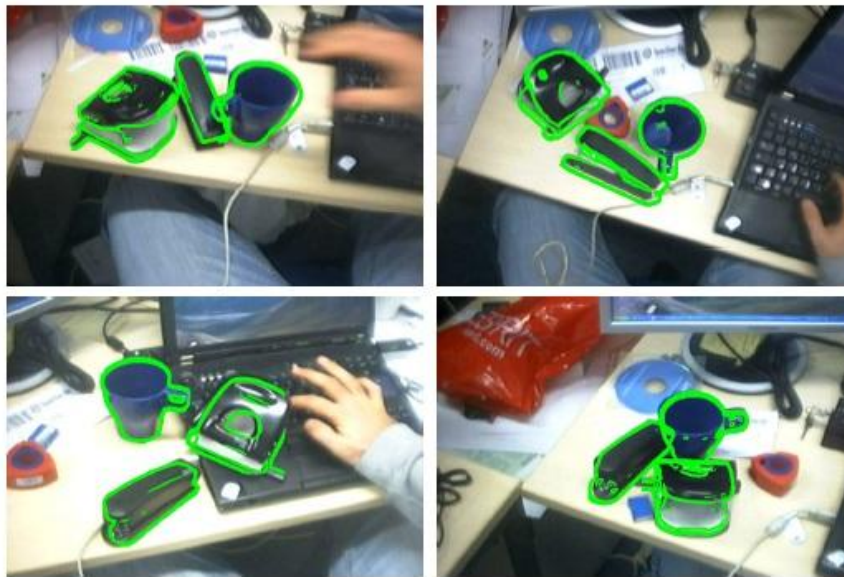
Figura 1.17 – Panoramica del funzionamento dell'algoritmo.

1.6.2. Features matching

L'associazione dei punti è una parte molto delicata e difficile poiché le fonti d'errore sono molteplici e anche piccole inesattezze spesso conducono a risultati molto differenti da quelli attesi.

L'assunzione generale consiste nel considerare che due immagini successive non siano molto diverse fra di loro e, quindi, l'intorno di una feature estratta non dovrebbe variare molto tra un frame e l'altro. Tuttavia, è molto probabile che un punto d'interesse di un'immagine abbia più di una possibile corrispondenza e diventi necessario scegliere l'accoppiamento migliore sulla base di qualche criterio.

Le fasi di matching e di tracking consistono nella ricerca di corrispondenze fra le features estratte da due immagini della stessa scena.

Nel primo caso la ricerca delle corrispondenze avviene tra due viste della stessa scena che differiscono per la posizione della telecamera. Un esempio di matching è l'operazione che si compie nella visione stereo per trovare correttamente le porzioni d'immagine equidistanti delle due telecamere[42].

Il Tracking, invece, è la ricerca delle corrispondenze tra immagini della stessa scena ripresa in due istanti di tempo successivi. Il tracciamento, quindi, considera, implicitamente, il problema del movimento di oggetti all'interno della scena, o reciprocamente, della telecamera, rispetto ad un riferimento fisso. Un esempio di tracking è la stima della posizione di un robot che si muove in un ambiente sconosciuto[34]. Nell'affrontare il problema delle corrispondenze tra features, esistono particolari vincoli che limitano e aiutano la ricerca delle giuste corrispondenze in relazione all'applicazione che si sviluppa e all'hardware impiegato.

L'applicazione realizzata col presente lavoro di tesi è stata sviluppata utilizzando un sistema mono-camera mobile nello spazio. La ricerca delle corrispondenze fra features avviene tra il frame corrente, acquisito durante il movimento della telecamera, e un' immagine, precedentemente salvata, dell'oggetto da individuare nella scena. Utilizzando le corrispondenze trovate viene poi identificata la trasformazione spaziale che caratterizza l'oggetto e, considerando le trasformazioni calcolate nei frame precedenti, si calcola la posizione dell'osservatore.
Il tracking ed il matching, benché siano due fasi distinte, si riconducono, quindi, alla stessa radice.
Entrambe le operazioni, infatti, ricadono in due filoni principali di tecniche: i metodi descriptor-based e quelli correlation-based.
Le tecniche basate sui descrittori estraggono un set di features da ogni frame acquisito e, successivamente, cercano di stabilire le corrispondenze tra i due insiemi di punti. Tali tecniche richiedono che vengano estratte le stesse features, in modo affidabile e consistente, in entrambi i frame: ovviamente ci sono problemi intrinseci dovuti al movimento, che farà uscire dalla scena inquadrata features estratte nel frame di partenza e, allo stesso tempo, favorirà l'ingresso di nuovi punti d'interesse a causa dell'acquisizione di nuove porzioni di sfondo.
Il grande svantaggio di tali tecniche è che gli errori nelle corrispondenze tendono ad essere molto grandi se l'algoritmo non garantisce una buona affidabilità.
I metodi basati sulla correlazione, invece, necessitano solamente dell'estrazione di un unico set di features dal frame di partenza. La posizione dei punti estratti nei frame seguenti viene trovata compiendo una ricerca su una finestra di dimensioni opportune, cercando il template che meglio si correla con quello attorno al punto nel primo frame. Lo svantaggio principale di questa tecnica è il tempo di calcolo, che dipende fortemente dalle dimensioni della finestra sulla quale si esegue la ricerca.

1.6.2.1. Il matching descriptor-based

Quando l'estrattore di features aggiunge all'informazione sulle coordinate del punto d'interesse anche un vettore contenente una descrizione del punto estratto, delle sue caratteristiche e del suo intorno, allora la ricerca delle corrispondenze può sfruttare tutta questa conoscenza sulle features per velocizzare notevolmente l'operazione di matching.
In questo tipo di matching viene utilizzato il noto metodo Nearest-Neighbour-Ratio[50]: dato un punto-feature F1 nella prima immagine, si trova la corrispondenza con un punto-feature F2 nella seconda immagine se F2 è il "vicino più vicino"(*nearest neighbour*) di F1 (nello spazio delle features) e se il rapporto tra la distanza di F1 da F2 e quella di F1 dal suo secondo vicino non eccede una soglia (**Fig. 1.18**).
L'affidabilità di questo tipo di matching migliora notevolmente attraverso la bi-direzionalità dell'operazione stessa: un match è valido se e solo se la feature F1, estratta dal primo frame, seleziona come corrispondenza la feature F2 nel secondo frame e, viceversa, quest'ultima sceglie come sua controparte la feature F1 stessa (*mutual consistency check*).

L'algoritmo di matching può essere ulteriormente migliorato sfruttando particolari vincoli in modo da velocizzare ancor di più il tempo di calcolo ma soprattutto per controllare un minor numero di features, diminuendo la possibilità di trovare dei falsi positivi, quei punti, cioè, che, pur non avendo una corrispondenza reale, vengono associati perché conformi ai vincoli posti, conducendo ad errori nella stima del movimento.
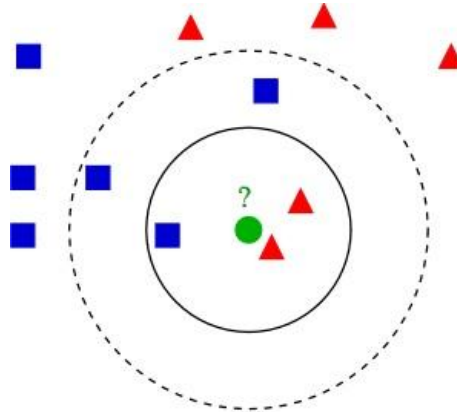
Figura 1.18 – Esempio di ricerca del nearest neighbour.

1.6.3. Features tracking

In letteratura esistono diversi algoritmi di tracking, tra i quali si può citare il KLT (Kanade-Lucas-Tomasi) feature tracker, che si basa sul precedente lavoro di Lucas e Kanade [43] e sullo sviluppo di Tomasi e Kanade [64]: le feature estratte dal Shi-Tomasi detector vengono inseguite per mezzo di un metodo che minimizza la differenza tra due finestre costruite attorno a ciascun punto. Tale tipo di algoritmo tuttavia richiede che l'acquisizione dei frame venga fatta in maniera ravvicinata nel tempo, in modo che immagini diverse si discostino minimamente; tutto ciò richiede uno sforzo computazionale molto elevato.

1.6.4. Stima della Camera Pose

Il passo finale dello schema in **Fig. 1.6** prevede l'utilizzo di algoritmi che, a partire dagli insiemi di punti 3D ottenuti attraverso le fasi precedenti, generino una stima della posizione dell'osservatore.

In letteratura sono state adoperate varie tipologie di algoritmi, assieme ai quali spesso sono state integrate tecniche di individuazione e rimozione dei punti che rappresentano valori anomali e che sono, quindi, distanti dalle altre osservazioni disponibili, detti outliers[18].

Sfruttando le trasformazioni prospettiche ed i parametri intrinseci della telecamera, come detto in precedenza nella sezione relativa alla calibrazione, è possibile passare dalle coordinate 2D nel piano immagine a coordinate 3D nel sistema di riferimento telecamera.

Capitolo 2
La Realtà Aumentata

La *Realtà Aumentata* (o AR: **Augmented Reality**) è una tecnologia che permette di intervenire in tempo reale su di un flusso di immagini, visualizzando su un display contenuti ed animazioni digitali con il fine di migliorare la qualità e la quantità delle informazioni ad esso correlate.
A differenza della Realtà Virtuale, il cui scopo è quello di realizzare una percezione realistica di un ambiente totalmente artificiale che un utilizzatore può esplorare interattivamente attraverso i propri sensi (vista, audio, tatto), la Realtà Aumentata si propone di arricchire la percezione visiva del mondo reale attraverso l'aggiunta di informazioni generate dal computer (testo, immagini sia 3D che 2D, suoni ecc..) alla realtà percepita dall'utente del sistema.
Il presente lavoro di Tesi sfrutta le caratteristiche e le potenzialità delle tecniche di Realtà Aumentata con l'obiettivo di guidare, monitorare e rendere efficiente l'esecuzione di operazioni di assemblaggio, disassemblaggio e manutenzione nell'ambito industriale.

2.1. Definizioni

Ad oggi esistono principalmente due definizioni ugualmente riconosciute di Realtà Aumentata.
Secondo la definizione proposta da R. Azuma[3] un sistema AR deve avere le seguenti proprietà:

- deve combinare il reale con il virtuale;
- deve essere eseguito interattivamente e in real time;
- deve allineare oggetti reali con quelli virtuali e viceversa.

La prima proprietà è quella fondamentale per un sistema AR. La seconda richiede che il sistema reagisca agli input dell'utente e si aggiorni in tempo reale. La terza proprietà distingue la realtà aumentata dal concetto più generale di mixed-reality: le immagini virtuali 3D devono essere allineate geometricamente agli oggetti reali nel mondo reale.
Come si può osservare in **Fig. 2.1**, riferendosi al *reality-virtuality continuum*, proposto da Milgram[49], la realtà aumentata si inserisce, appunto, tra il mondo reale e la realtà virtuale.
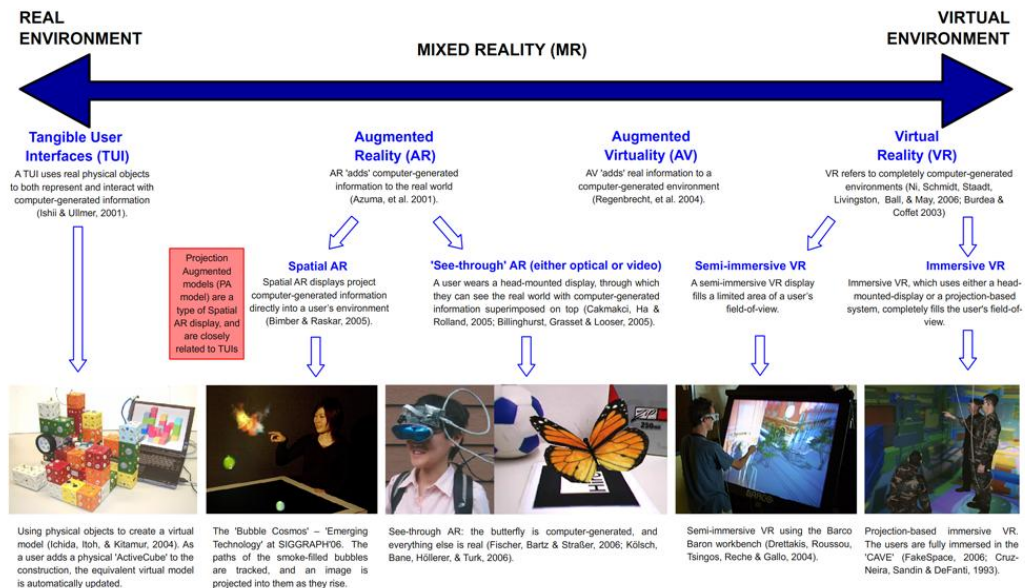
Figura 2.1 – Reality-Virtuality Continuum di Milgram.

2.2. Cenni Storici

Il primo sistema AR completamente funzionante risale al 1968 quando Ivan Sutherland e il suo team costruirono un display see-through head mounted (HMD) con tracking meccanico, mediante il quale l'utilizzatore del dispositivo poteva vedere le informazioni virtuali generate dal computer sovrapposte agli oggetti fisici.

Negli anni ‘70 e ‘80 la realtà aumentata è diventata un'area di ricerca in diverse istituzioni americane come l'U.S. Air Force's Armstrong Laboratory, il NASA Ames Research Center, il Massachusetts Institute Of Technology e l'università del North Carolina, a Chapell Hill.
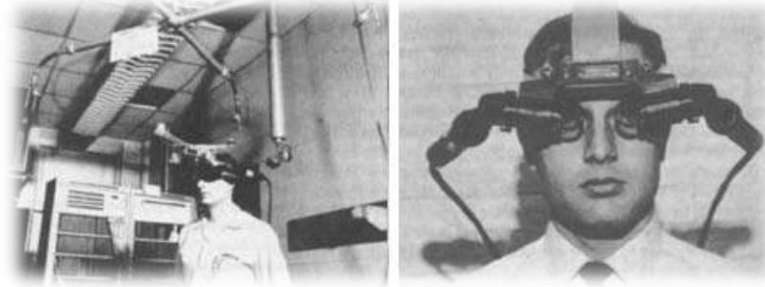
Figura 2.2 – Il primo Head Mounted display della storia.

Fu solo nel 1990, nei centri di ricerca della Boeing Corporation, che l'idea di sovrapporre la grafica generata dal computer al mondo reale ha ricevuto il suo nome attuale. Due ricercatori della Boeing cercando di semplificare l'assemblaggio dell'impianto elettrico per le persone addette alla costruzione degli aerei, decisero di sovrapporre le informazioni virtuali alle immagini reali. A metà degli anni ‘90, grazie alla continua miniaturizzazione dei componenti elettronici e alla maggiore potenza di essi, cominciarono ad essere implementati i primi dispositivi AR mobili (MARS: Mobile Augmented Reality System), composti da un computer mobile, dispositivi di tracking e un HMD.
Uno dei primi prototipi fu il Columbia *Touring Machine* (**Fig. 2.3**), che presentava informazioni grafiche 3D ai visitatori del campus, relativamente agli edifici che l'utente stava attualmente guardando attraverso l'HMD[29]. Nell'ottobre del 1998 si tennel'*International Workshop on Augmented Reality*, la prima conferenza sulla realtà aumentata, presso San Francisco.

Figura 2.3 – MARS Touring Machine.

Grazie al continuo sviluppo tecnologico, la realtà aumentata ha ormai applicazioni nei più disparati settori, commerciali e di ricerca. Ad oggi, è possibile trovare sistemi AR funzionanti anche su tablet e telefoni cellulari.

2.3. Modalità e strumenti per l'interazione

In una architettura AR sono sempre presenti quattro componenti: display, dispositivi di tracking, dispositivi per acquisizione di immagini ed un processore per generare la grafica tridimensionale.
Nel seguito del capitolo si approfondiranno i primi due componenti, poiché quelli di maggior impatto nell'architettura di un sistema di realtà aumentata.

2.3.1. Display

I display impiegati nella realtà aumentata sono hardware che impiegano componenti ottiche ed elettroniche per generare immagini nella traiettoria visiva tra gli occhi dell'osservatore e l'oggetto fisico. La **Fig. 2.4** mostra una classificazione dei vari tipi di display in funzione della posizione occupata rispetto all'osservatore ed all'oggetto osservato[55].
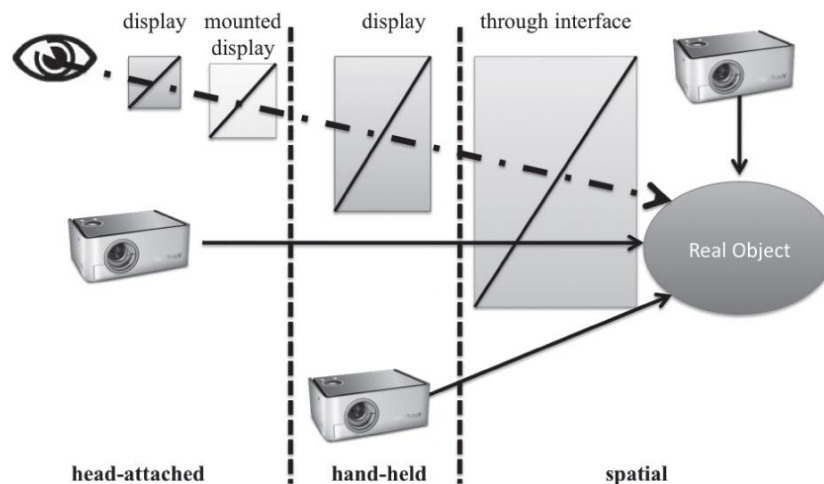
Figura 2.4 – Classificazione dei Display per AR.

I *near-eye displays* (o *head-attached displays)*, come, per esempio, *retinal displays*, *head-mounted displays* ed *head-mounted projectors*, devono essere direttamente indossati dall'utente. Altri tipi di display sono invece *hand-held*, cioè possono essere tenuti tra le mani dall'osservatore, come per esempio telefoni cellulari, palmari e tablet. Infine la terza categoria comprende quei display che sono allineati spazialmente tra l'osservatore e l'oggetto reale. Questi generalmente fanno uso di videoproiettori ed apposite superfici sulle quali visualizzare le immagini[4].

Sono stati individuati diversi fattori che devono essere considerati nella scelta di quale tecnologia utilizzare per visualizzare le informazioni:

- *Latenza*: quantità di tempo che intercorre da quando l'utente si muove a quando l'immagine finale viene visualizzata sul display;
- *Risoluzione* della scena reale e distorsione: risoluzione con la quale la realtà viene presentata all'utente e quali cambiamenti sono introdotti dall'ottica;
- *Campo visivo*: porzione di vista permessa dal display all'utente;
- Fattori di *costo*: in certe applicazioni sono richieste ottiche complesse che innalzano i costi.

2.3.1.1. Near-Eye Displays

Nei near-eye display, il sistema di visualizzazione è indossato dall'utente sulla testa. In base alla tecnologia di generazione dell'immagine esistono tre diverse tipologie di near-eye display:

- Head-mounted displays;
- Retinal displays;
- Head-mounted projectors.

Gli *head-mounted displays* (HMDs) sono attualmente i sistemi più utilizzati nelle applicazioni di realtà aumentata. Un *see-through HMD* è un dispositivo utilizzato per combinare il reale ed il virtuale. I *closed-view HMD* di tipo standard non permettono nessuna visione diretta della realtà. Diversamente invece un display *see-through* permette all'utente di vedere il mondo reale con oggetti virtuali sovraimpressi attraverso tecnologie ottiche o video.

Un *optical see-through HMD* (**Fig. 2.5**) funziona posizionando dei combinatori ottici di fronte agli occhi dell'utente. Questi combinatori sono parzialmente trasmissivi: attraverso di essi l'utente può vedere il mondo reale. Questi combinatori sono anche parzialmente riflessivi, in modo tale che l'utente possa vedere le immagini virtuali di fronte agli occhi.
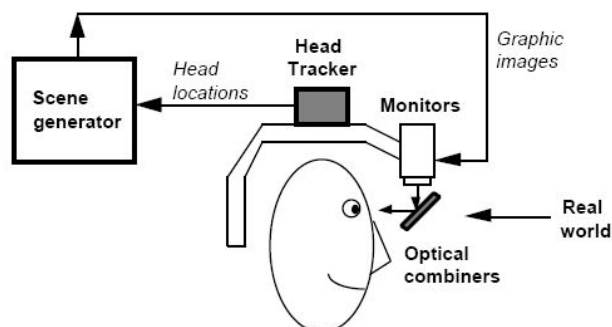
Figura 2.5 – Optical See-Through Display.

Un *video see-through HMD* (**Fig 2.6**) funziona combinando un *closed-loop HMD* con una o più *head-mounted video cameras*, le quali forniscono all'utente le immagini del mondo reale. Il video fornito da queste videocamere viene combinato con le immagini generate dal computer, unendo quindi le immagini reali con quelle virtuali. Il risultato viene inviato ad un monitor posto di fronte agli occhi dell'utente.
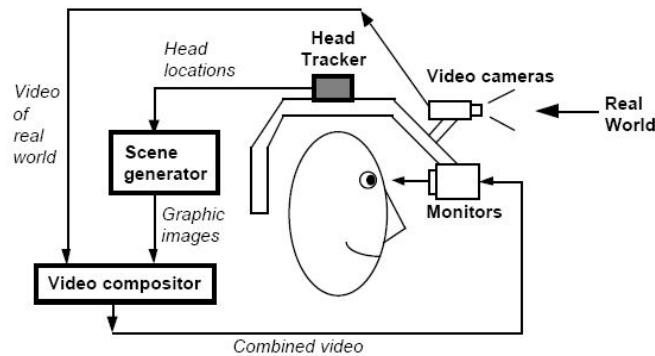
Figura 2.6 – Video See-Through Display.

L'approccio ottico ha i seguenti vantaggi rispetto a quello video[74]:

- **Semplicità.** Unire le immagini virtuali e reali è più semplice ed economico rispetto alla tecnologia video. L'approccio ottico ha infatti un solo video stream di cui preoccuparsi: le immagini grafiche generate dal computer. Il mondo reale infatti è visto direttamente attraverso i combinatori ottici e, generalmente, il ritardo è di pochi nanosecondi.
  Nell'approccio video invece devono essere gestiti due flussi video: quello proveniente dalle videocamere e quello delle immagini virtuali. Questi sistemi hanno un ritardo in termini di decine di millisecondi.
- **Risoluzione**. Nella tecnologia video, la risoluzione con la quale l'utente vede sia l'immagine reale che quella virtuale è limitata alla risoluzione del display. Anche l'*optical see-through* visualizza le immagini virtuali alla risoluzione del display ma l'immagine reale non viene degradata in quanto l'utente la vede direttamente con i propri occhi.
- **Sicurezza.** Il *video see-through HMD* è essenzialmente un *closed-view HMD* modificato. In caso di mancata alimentazione l'utente è completamente cieco ed in certi scenari questo può essere un pericolo. Se invece l'alimentazione fosse rimossa da un *optical-see through HMD* l'utente sarebbe ancora capace di vedere il mondo reale.
- ***No eye offset*.** Con il *video see-through*, la vista del mondo reale viene fornita attraverso videocamere. In molte configurazioni le videocamere non sono poste esattamente dove si trovano gli occhi dell'utente, creando quindi un offset tra le videocamere e gli occhi delle persone. Questa differenza genera uno scostamento tra quello che l'utente vede rispetto a quello che si aspetterebbe di vedere.

In contrasto la tecnologia video offre i seguenti vantaggi rispetto a quella ottica:

- **Larghezza del campo visivo.** Nei sistemi ottici le distorsioni sono funzioni della distanza radiale dall'asse ottico. Più si guarda lontano dal centro di vista, più è elevata la distorsione. Un'immagine digitale catturata attraverso un sistema ottico che presenta distorsione può essere corretta attraverso tecniche di elaborazione delle immagini.
- **Ritardo di visualizzazione.** Con l'approccio video è possibile ridurre o evitare i problemi causati dalla differenza temporale con cui sono processate le immagini reali e quelle virtuali. L'*optical see-through HMD* permette una visione istantanea del mondo reale ma l'immagine virtuale viene visualizzata con un certo ritardo. Con l'approccio video è possibile ritardare il video del mondo reale per uguagliare il ritardo dello stream dell'immagine virtuale.
- E' più facile uguagliare il colore e la brillantezza delle immagini reali e virtuali.

I *retinal displays* (RDs) (**Fig. 2.7**) utilizzano laser a semiconduttori a bassa potenza per proiettare le immagini direttamente nella retina dell'occhio umano.
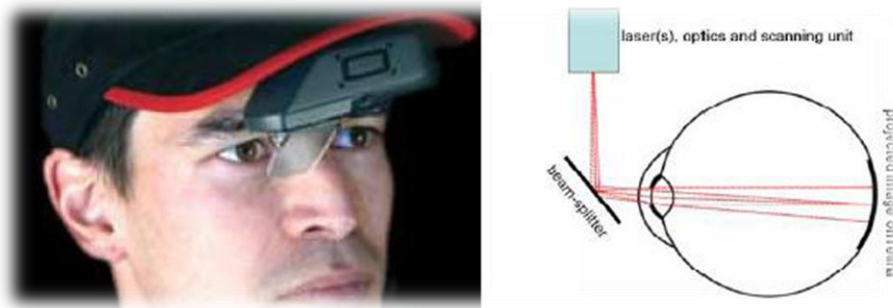
Figura 2.7 – HMD Retinal display e schema di funzionamento.

I principali vantaggi di questi dispositivi sono:

- immagine più luminosa e più definita;
- campo visivo maggiore rispetto alle tecnologie basate sull'utilizzo di schermi;
- basso consumo energetico.

Al contrario i principali svantaggi sono:

- immagini monocromatiche;
- la lunghezza focale risulta fissa;
- non esistono versioni stereoscopiche di RDs.

Questi dispositivi sono adatti per applicazioni mobili in spazi all'aperto dove non si può avere un controllo sulla luce ambientale e dove è fondamentale tenere sotto controllo il consumo energetico.

Gli head-mounted projectors (HMPs) sono dispositivi che indirizzano il fascio di luce proiettata con un divisore di fascio ottico in modo tale che l'immagine sia diretta verso superfici retroriflettenti, collocate di fronte all'utente
Gli HMPs forniscono un campo visivo maggiore rispetto agli HMDs ed evitano distorsioni dovute ad errori di parallasse. Tuttavia la luminosità delle immagini dipende essenzialmente dall'illuminazione dell'ambiente circostante e gli attuali modelli di HMPs sono scomodi, ingombranti e troppo pesanti per essere utilizzati in ambiti professionali.

Figura 2.8 – Realtà Aumentata Projection-based (Volkswagen).

2.3.1.2. Hand-Held Displays

Smartphone e Tablet PC sono esempi di *hand-held displays* (HHDs). Questi dispositivi, inglobando in un'unica apparecchiatura processore, memoria, display e dispositivo di acquisizione video, permettono di realizzare sistemi portatili senza fili (**Fig. 2.9**).
L’approccio di tali dispositivi è *video see-through*: la videocamera presente nell'apparecchio cattura un flusso video dell'ambiente e lo visualizza sul display arricchendone il contenuto di informazioni con immagini generate dal processore grafico.
I principali vantaggi degli HHDs sono rappresentati dalla semplicità d'uso, dalla compattezza e dalla limitata invasività. Tuttavia la loro contenuta capacità di calcolo

Figura 2.9 – Applicazione di Realtà Aumentata su tablet PC.

può causare un eccessivo ritardo nella sovrapposizione di immagini grafiche sull'immagine reale e la piccola dimensione dello schermo offre un limitato campo visivo.

2.3.2. Sistema di tracciamento

Nella realtà aumentata l'operazione di tracciamento (tracking) si rende necessaria per ottenere la posizione e l'orientamento nello spazio dell'utente ed allineare correttamente (*registration*) l'immagine virtuale a quella reale.
Quando l'utente cambia il suo punto di vista l'immagine generata artificialmente deve rimanere allineata con la posizione e l'orientamento dell'oggetto reale osservato e quindi è necessario che il sistema di tracciamento aggiorni in tempo reale la nuova posizione dell'utente. Il tracciamento può essere effettuato inseguendo diverse parti del corpo umano, come per esempio le mani, la testa o anche l'intero corpo.
Di solito, l'operazione di tracking per la Realtà Aumentata si concentrano nella determinazione di posizione ed orientamento della testa dell'utente. In questo caso si parla di *head tracking*.
In commercio, ed in letteratura, sono presenti numerosi differenti dispositivi di tracking, le cui varie caratteristiche possono essere riassunte nelle seguenti [29]:

- Risoluzione: la misura della più piccola unità spaziale che il sistema di tracciamento può misurare;
- Accuratezza: intervallo entro il quale la posizione riportata può essere considerata corretta;
- Reattività del sistema, cioè:
  - Frequenza di campionamento: velocità con la quale il sensore controlla i dati, generalmente espressa come frequenza (Hz);
  - *Data rate*: quante volte viene rilevata la posizione in un secondo, espressa in frequenza (Hz);
  - *Update rate*: velocità con cui il sistema di tracking riporta la nuova posizione calcolata al computer, anche questa espressa in frequenza (Hz);
  - Latenza: ritardo tra il movimento dell'oggetto o dell'utente con il sistema di tracking e il calcolo della nuova posizione. Viene espresso in millisecondi (ms). Generalmente questo ritardo deve essere inferiore ai 60 ms.
- Range Operativo: misura dello spazio in cui il tracciamento riesce a funzionare. Ad esempio alcuni sistemi utilizzano segnali emessi da sorgenti per calcolare la posizione. Più il sensore si allontana dalla sorgente, più il segnale ricevuto si attenua. E' quindi chiaro che esisterà una distanza massima oltre la quale non sarà possibile calcolare la posizione, a causa dell'assenza del segnale dalla sorgente.
- Costi: i costi variano a seconda della complessità del sistema di tracking e dell'accuratezza;
- Trasportabilità: sono importanti anche grandezza e peso del sistema, qualora debba essere anche indossato dall'utente.

La misura della posizione e dell'orientamento effettuate dai sistemi di tracciamento è relativa ad una posizione e ad un orientamento di riferimento e viene effettuata mediante l'utilizzo di sorgenti di segnali e sensori in grado di intercettarli. A seconda di dove sono disposte queste due componenti i meccanismi di tracking vengono suddivisi in tre categorie:

- *Inside-in*: i sensori e le sorgenti si trovano entrambi sull'oggetto di cui si vuole misurare la posizione. Di solito questo tipo di meccanismo non è in grado di fare misurazioni assolute rispetto ad un sistema di riferimento esterno, ma solamente relativa ad uno stato iniziale.
- *Inside-out*: i sensori sono sul corpo mentre le sorgenti si trovano all'esterno in posizioni prefissate. Questi meccanismi sono in grado di fornire posizioni riferite ad un sistema di riferimento assoluto.
- *Outside-in*: le sorgenti si trovano sull'oggetto, mentre i sensori sono posti in punti esterni prefissati. Come i precedenti, anche questa tipologia di meccanismi è in grado di fornire coordinate assolute. Questo approccio è sicuramente il meno intrusivo, poiché non prevede che l'utente sia equipaggiato con sensori, che possono essere voluminosi e necessitare di alimentazione[51].

Esiste anche un altro metodo di suddivisione dei dispositivi di tracking che li raggruppa in due categorie:

- *Active-target*: sistemi che includono sorgenti di segnali artificiali. Esempi di tali sistemi sono quelli che fanno uso di segnali magnetici, ottici, radio e acustici.
- *Passive-target*: sistemi che utilizzano l'ambiente o segnali che esistono in natura. Esempi sono bussole, girobussole e sistemi ottici che utilizzano *fiducial markers* o caratteristiche naturali dell'ambiente (*natural features*).

Il **tracking inerziale** utilizza giroscopi e accelerometri per determinare posizione e orientamento nello spazio dei dispositivi inerziali impiegati. Tale processo è soggetto al problema della deriva, per cui necessita frequenti calibrazioni dell'apparecchiatura.
I sistemi di **tracking acustico** basano il loro funzionamento sull'emissione di onde ultrasoniche alla frequenza di 20kHz, valore al di sopra delle frequenze riconosciute dall'apparato uditivo umano.
I pregi principali di questi sistemi sono la leggerezza e l'indipendenza da disturbi dovuti ad interferenze magnetiche. Per contro, la forte sensibilità alle interferenze acustiche e la necessità di avere sempre la visuale libera tra trasmettitori e ricevitori ne limita l'affidabilità.
Il **tracking magnetico** si basa sulla misura del campo magnetico e utilizza sia onde a bassa frequenza che pulsate. I principali svantaggi connessi all'impiego di questa tecnologia riguardano la sensibilità alle interferenze magnetiche causate da onde radio e superfici metalliche. Infine l'accuratezza di misura diminuisce con l'aumentare della distanza.

I sistemi **meccanici** di tracking misurano gli angoli e le lunghezze tra i giunti del dispositivo ed utilizzano una configurazione base come posizione iniziale rispetto alla quale misurare i movimenti degli oggetti puntati. Questo tipo di tecnologia non è sensibile alle interferenze ed è in grado di fornire misure precise, sia sulla posizione che sull'orientamento. Per contro, i sistemi di questo tipo risultano piuttosto invasivi, ingombranti e consentono spazi di lavoro limitati.
Risulta evidente che i dispositivi di tracking basati sulla visione artificiale siano una soluzione economica ed efficace per risolvere il problema del tracciamento. Questi sistemi sfruttano la luce per stimare la posizione nello spazio dell'oggetto da inseguire. Il sistemi di tracking di tipo ottico assicurano velocità di elaborazione adeguate e non pongono limiti sullo spazio di osservazione.
Il vantaggio di usare la visione per la realizzazione di un sistema di localizzazione e tracciamento risiede sostanzialmente nella quantità di informazioni che è possibile ottenere anche senza l'impiego di hardware speciali e costosi, come ad esempio sensori di posizione, fra l'altro scomodi da posizionare sull'oggetto da tracciare. Sfortunatamente, estrarre informazioni affidabili e precise dalle immagini non è un'impresa facile e lo diventa ancor meno se dalla sequenza di immagini bisogna estrarre anche la terza dimensione necessaria al calcolo della posizione 3D dei soggetti tracciati. Le tecniche di visione 3D per effettuare misure e riconoscimento di oggetti nello spazio, infatti, rappresentano un argomento di ricerca attivo ed in continua evoluzione.
Nel capitolo 1 sono stati analizzati lo stato dell'arte e la metodologia dei più recenti algoritmi di tracking basato sulla visione artificiale. Nel paragrafo successivo si introdurrà, invece, il concetto di markerfield .

Figura 2.10 – Esempio di applicazione di Realtà aumentata con marker fiduciali (ARToolkit).

In semplici applicazioni di realtà aumentata, ogni singolo oggetto virtuale è associato ad uno specifico marker, precedentemente calibrato, e viene sovrapposto ad esso quando viene individuato dall'algoritmo di marker detection.
In applicazioni di realtà aumentata più complesse, un solo marker non è sufficiente per definire un sistema di riferimento per visualizzare modelli virtuali all'interno della scena.

Nel caso in cui, per esempio, deve essere effettuato il tracking di oggetti ingombranti o della posizione di una persona che si muove all'interno di un'area di grandi dimensioni, appare evidente come non sia possibile affidarsi ad un unico marker ben visibile, ma sia, invece, necessario ricorrere ad un insieme di markers, definiti rispetto ad un'origine globale, detto *marker field (o marker set*)[7]. In questo modo gli oggetti virtuali possono essere visualizzati e collocati all'interno della scena nei casi in cui un marker non sia rilevabile, perché occluso o fuori dal campo visivo.
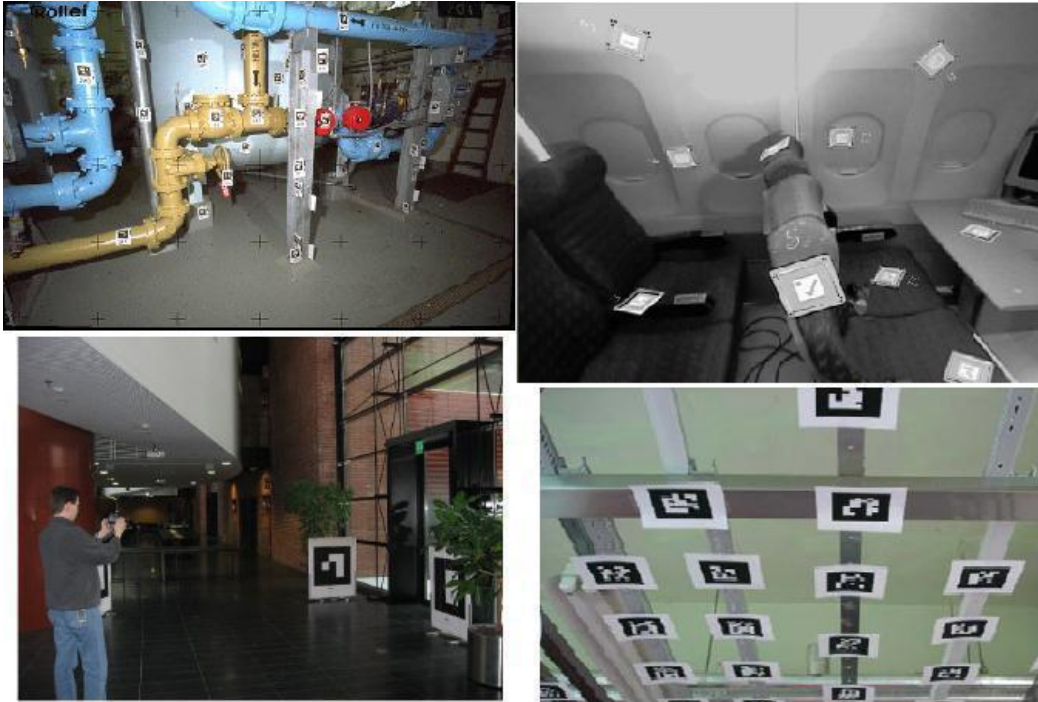
Figura 2.11 – Esempi di Markerfield complessi.

2.3. Applicazioni della Realtà Aumentata

Grazie alla peculiare caratteristica di arricchire il contenuto informativo dell'utente che fruisce di un dispositivo di visualizzazione, la AR permette un notevole miglioramento della qualità e dell'efficacia dell'interazione uomo-macchina.

Sovrapponendo modelli virtuali ad una scena reale osservata, la AR consente di sintetizzare e visualizzare informazioni addizionali direttamente nel contesto di una specifica area di interesse sulla quale si focalizza l'attenzione dell'utente del sistema di Realtà Aumentata.

Applicazioni di apprendimento guidato per la formazione del personale e di orientamento e guida di utenti in aree a loro sconosciute sono solo alcuni interessanti casi d'utilizzo di un sistema AR.

Nel seguito, si presenteranno in breve numerosi casi applicativi nei più disparati settori dell'attività umana.

2.3.1. Assemblaggio, Manutenzione e Riparazione

Un ambito in cui la realtà aumentata può risultare molto utile è quello industriale dove può essere utilizzata per assistere gli operatori addetti all'assemblaggio di componenti meccanici[15], oppure i tecnici durante le attività di manutenzione o riparazione di determinati macchinari[53]. Molte industrie, sempre più spesso, progettano e sviluppano al computer, in modo da sostituire i prototipi fisici con prototipi virtuali per il packaging e per effettuare dei test sicuri. L'utilità di questa preferenza progettuale risulta particolarmente evidente nel caso di produzione di aerei ed auto, dove i prototipi fisici sono costosi e il time-to-market è un fattore determinante. Le istruzioni per assemblare o riparare macchinari infatti, sono più semplici da capire se sono disponibili sotto forma di immagini 3D e non come manuali cartacei: ad esempio possono essere generate istruzioni su un HMD, sovrapposte al dispositivo reale, che mostrano con delle animazioni (ad esempio frecce che indicano la posizione ed il verso in cui inserire un pezzo) i passi da compiere per l'assemblaggio o la riparazione (**Fig.2.12**). In questo modo può essere notevolmente velocizzato il processo di produzione e manutenzione e ridotta anche la probabilità di errore da parte di un operaio.
Il presente lavoro di tesi si inserisce, appunto, nel presente contesto.
La Boeing, come già accennato precedentemente, è stata la prima ad utilizzare l'AR per questo scopo, sviluppando un dispositivo che permetteva di assistere gli operai nell'assemblaggio del circuito elettrico degli aerei visualizzando informazioni 3D su un display.
Negli ultimi anni sono nati molti progetti riguardo l'AR in ambito industriale come per esempio ARVIKA, AugAsse, PLAMOS, di cui si parlerà in un capitolo successivo.

Figura 2.12 – Esempio di utilizzo della realtà aumentata in ambito industriale.

2.3.2. Medicina

La realtà aumentata nel campo della medicina può venire in aiuto dei chirurghi in fase di studio di un intervento su un paziente. Mediante sensori come il *Magnetic Resonance Imaging* (MRI) o il *Computed Tomography scan* (CT) è, infatti, possibile ottenere immagini tridimensionali del paziente[38]. Queste immagini, visualizzate su un display HMD, possono essere allineate e sovrapposte alla vista reale del paziente. Questo risulta utile in caso di un intervento di chirurgia mini invasiva permettendo al chirurgo di avere una vista interna del corpo mediante immagini tridimensionali sovrapposte a quelle reali, senza la necessità di ricorrere a profonde incisioni (**Fig.2.13**).

I medici possono utilizzare la realtà aumentata anche durante il loro giro di visita ai pazienti, visualizzando informazioni su di essi direttamente su un display invece di leggere le cartelle cliniche.
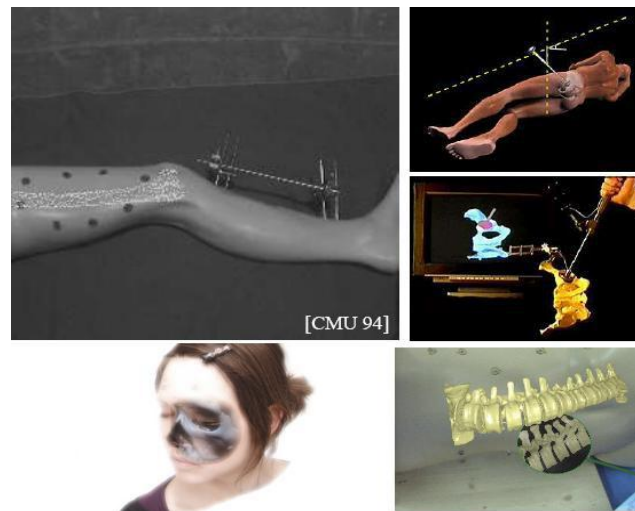
Figura 2.13 – Applicazioni medicali della Realtà Aumentata.

2.3.3. Archeologia e Architettura

La realtà aumentata, in questo ambito, può essere utilizzata, per esempio, per visualizzare in modo tridimensionale, nella realtà, i progetti di edifici o ponti che verranno poi costruiti in un particolare sito, oppure per ricostruire siti archeologici o per visualizzare antichi edifici nella loro locazione originale[81].
Ne è un esempio il progetto *ARCHEOGUIDE* sponsorizzato dalla comunità europea che permette di ricostruire siti archeologici, ricostruendo in 3D gli edifici mancanti e visualizzando le parti mancanti di un manufatto su un HMD (**Fig. 2.14**).

2.3.4. Turismo

E' possibile utilizzare la realtà aumentata non solo per guidare un turista verso una particolare destinazione, ma anche per visualizzare informazioni.
Invece di avere informazioni (storiche e non) con una guida turistica cartacea, il turista può visualizzarle sul proprio smartphone mentre sta guardando l'edificio o il luogo d'interesse.
Ad esempio il *Touring Machine* della Columbia University permette di guidare un turista all'interno del campus universitario e di ottenere informazioni 3D relative ad un certo edificio su un HMD.

Figura 2.14 – Esempi di AR in architettura ed archeologia (progetto ARCHEOGUIDE).

2.3.5. Intrattenimento

La AR può essere utilizzata poi per combinare attori reali con ambienti virtuali, riducendo così i costi di produzione di un film. L'attore si muove davanti ad un grande schermo blu, mentre una telecamera mobile registra la scena. Poich la posizione della telecamera in ogni istante è nota, così come la posizione dell'attore, è possibile allora far muovere l'attore in un ambiente 3D. Anche i videogames possono far uso della realtà aumentata: esempi sono *ARQuake*[80], sviluppato dal *Wereable Computer Lab* alla *University of South Australia* che permette di combattere contro mostri virtuali camminando in un ambiente reale, oppure *The Eye Of Judgement*[76] per Sony Playstation 3.

2.3.6. Campo Militare

La Realtà Aumentata, così come per molte altre tecnologie, ha avuto un rapido sviluppo tecnologico grazie all'interesse della ricerca nel campo militare.
I piloti dei caccia, ad esempio, utilizzano AR nei propri caschi, per avere informazioni sulla missione in tempo reale, senza dover distogliere lo sguardo dalla guida del velivolo.
Molte missioni militari hanno luogo in territori sconosciuti ai soldati. La realtà aumentata può essere utilizzata per mostrare una mappa tridimensionale del campo di battaglia, fornendo informazioni aggiuntive, come, ad esempio, nomi delle strade, posizione di edifici strategici (ospedale, stazione di comunicazione), la posizione dei soldati nemici, il tutto senza distogliere il soldato dalla vista reale dell'ambiente ed evitare quindi situazioni pericolose per la propria vita.

Figura 2.15 – F35 Helmet mounted display system.

2.4. Problematiche dell’interazione uomo-macchina

I principali limiti al realismo delle applicazioni di Realtà Virtuale ed Aumentata sono legati al ritardo nella risposta alle azioni degli utenti ed alla non corretta sincronizzazione degli eventi.
Le cause di questi effetti indesiderati sono da imputare al ritardo dei componenti e dell'architettura nel suo complesso. È possibile classificare queste latenze come segue[30]:

- ritardo del sensore: tempo che intercorre tra l'azione dell'utente ed il segnale di uscita dal sensore;
- ritardo della rete: ritardi dovuti al trasporto del segnale;
- ritardo computazionale: ritardo legato al calcolo degli algoritmi;
- ritardo di rendering: tempo trascorso per generare la grafica/audio dal motore di rendering;
- ritardo di sincronizzazione: il tempo in cui i dati sono in attesa prima di essere processati;
- ritardo del sistema operativo: non tutti i sistemi operativi sono adatti per applicazioni real time, in quanto l'esecuzione del codice può essere ritardata a causa di altri thread del sistema operativo;
- ritardo del display/attuatore: tempo che intercorre tra il segnale di input e la visualizzazione o reazione dell'attuatore.

Il tempo complessivo tra l'intervento da parte dell'utente e la risposta del sistema è definito *latenza end-to-end*.
Esistono in letteratura molti studi sul ritardo ammissibile in ambienti di realtà virtuale e aumentata. Per un feedback visivo di un movimento della mano, la latenza end-to-end deve essere al massimo di 100 ms, altrimenti causa nell'operatore movimenti lenti, prudenti e imprecisi[19].
In letteratura sono presenti numerosi studi sull’argomento. Molti sono stati effettuati utilizzando un HMD (Head Mounted Display) in una stanza chiusa.
Si è osservato che ritardi nella risposta compresi tra i 100 ed i 200 ms portino ad una percezione instabile dell'ambiente[2] e che una latenza di 50 ms provochi negli utenti una sensazione soggettiva di non presenza all'interno dell'ambiente, mentre a 90 ms di latenza si verifica un aumento statistico del ritmo cardiaco. Si è osservato[46] inoltre, che latenze superiori ai 120 ms siano risultate inaccettabili.
Per quanto riguarda invece il frame rate, si consiglia almeno il valore di 10 fps per fornire all'utente un'esperienza coinvolgente[38], anche se è auspicabile che il flusso di immagini abbia una frequenza attorno ai 20 fps, in modo che l'occhio umano possa percepire un flusso video fluido.

Capitolo 3
Analisi dello stato dell’arte delle applicazioni in ambito industriale per assistenza alle operazioni

Le attività di manutenzione e riparazione rappresentano un campo di applicazione interessante e pieno di opportunità per la realtà aumentata. La maggior parte di tali attività sono svolte da personale qualificato e preparato che esegue procedure stabilite da una documentazione tecnica in un ambiente relativamente statico e prevedibile. Queste procedure sono generalmente organizzate in sequenze di operazioni elementari da svolgere su particolare componenti in una posizione pre-determinata. L’insieme di queste ed altre caratteristiche permettono lo sviluppo di numerosi sistemi e tecnologie che possano assistere un tecnico durante lo svolgimento delle attività manutentive.
Sia gli impianti tecnologici che i prodotti industriali richiedono servizi di manutenzione globale, rapidi ed appropriati. Purtroppo, troppo spesso il livello di competenza degli operatori sul campo di tali sistemi non si dimostra adeguato ed al passo con il continuo mutamento tecnologico, e pertanto si richiede un continuo aggiornamento da parte del personale addetto e della manualistica di assistenza, non sempre reperibile o di facile comprensione.
Da un punto di vista psico-fisico, la manutenzione è un’attività dispendiosa e stressante, che richiede movimenti continui del collo e della testa nel passaggio da un’operazione ad un’altra. Le operazioni elementari richiedono, inoltre, un enorme sforzo cognitivo perché necessitano, prima dell’esecuzione, dell’identificazione dei vari

componenti all'interno di un modello dell'ambiente in cui si opererà realmente, quale quello rappresentato nei manuali di istruzione o nelle procedure di addestramento.
Questo problema è tanto maggiore quanto più specifico e complesso è il compito operativo del tecnico che esegue la manutenzione, come ad esempio nei casi industriali o aerospaziali.
Le attività di manutenzione in sistemi complessi, inoltre, comprendono un'ampia casistica di operazioni che include anche l'interazione con oggetti sconosciuti, distribuiti casualmente nello spazio operativo. Un altro fattore da tenere in conto, durante la progettazione e l'esecuzione dell'attività di manutenzione, è la difficoltà motoria dell'addetto, che potrebbe trovarsi ad operare in ambienti scomodi o nei quali la mobilità sia ridotta[26].
Il presente lavoro di Tesi è finalizzata alla realizzazione di un sistema che permetta di gestire e risolvere alcune delle criticità evidenziate.

3.1. "Manualistica Aumentata" - L'evoluzione del "manuale di istruzioni"

Il manuale di istruzioni è uno strumento, cartaceo o elettronico, che ha lo scopo di spiegare il funzionamento di un prodotto, di un dispositivo o di un servizio, avvalendosi di immagini sintetiche, linguaggio tecnico ed indice analitico. Può rivolgersi a diverse categorie di utilizzatori e, pertanto, può essere redatto con differenti finalità specifiche.
Il tipo di supporto maggiormente utilizzato è quello cartaceo, sebbene la diffusione di personal computer e dispositivi tascabili abbia notevolmente diffuso l'impiego di manuali elettronici. Questi manuali permettono di ridurre gli svantaggi di pesi ed ingombri dei manuali cartacei e consentono, attraverso rimandi ipertestuali, una lettura non lineare e l'accesso ad aggiornamenti e contenuti multimediali.
Spesso, però, modi espressivi poco chiari o scadenti metodi didattici possono indurre l'utilizzatore del manuale ad abbandonarne la lettura e preferire tentativi pratici, seppur maldestri.
Può succedere anche che un manuale fornito con un prodotto non sia specifico di quel prodotto, ma dell'intera serie: questo porta a far confusione tra l'aspetto esterno dell'oggetto e le figure del manuale oppure tra le caratteristiche descritte e quelle realmente presenti nel prodotto.
L'impiego della Realtà Aumentata si inserisce in questo contesto per cercare di risolvere questi ed altri problemi di carattere pratico che spesso si affrontano durante la consultazione di manualistica tecnica.

Uno dei primi studi sull'efficacia dell'AR applicata all'assemblaggio è stato realizzato da **Baird** e **Barfield**[6] nel 1998. L'obiettivo della ricerca era quello di valutare la fattibilità di un sistema indossabile (*wearable*) e mobile di Realtà Aumentata e confrontarne l'efficacia rispetto ai metodi tradizionali.

L'esperimento proposto dagli autori consisteva nel monitorare e confrontare come un gruppo di utenti eseguiva diverse operazioni manuali di assemblaggio di componenti elettronici, avvalendosi di diversi mezzi a supporto dell'esecuzione.
Le istruzioni venivano presentate con differenti tecnologie: un manuale d'istruzioni cartaceo, istruzioni di aiuto visualizzate su un computer (CAI: Computer Aided Instruction), e due tipi di head mounted display (opaque HMD e see-through HMD).
L'esperimento è stato condotto su un campione di 15 persone. Ogni soggetto, prima dell'esperimento, ha partecipato ad una sessione di formazione sull'assemblaggio dei vari componenti (**Fig. 3.1**) su di una scheda madre e ad una sessione pratica in cui veniva chiesto di installare ogni componente nel rispettivo slot, per acquisire praticità sull'operazione da compiere.
In modo casuale, ad ogni utente è stato assegnato un compito diverso. Ogni operazione è stata svolta impiegando in sequenza i 4 diversi supporti, misurando il tempo di esecuzione e gli errori compiuti.

Figura 3.1 – Componenti elettronici utilizzati nell'esperimento.

I **risultati** dell'esperimento hanno indicato che l'utilizzo della Realtà Aumentata risulta più efficace nelle applicazioni di apprendimento guidato rispetto ai mezzi tradizionali di supporto.
Il tempo medio di esecuzione delle operazioni, svolte con il supporto cartaceo, è stato di 197 secondi. Avvalendosi del supporto degli altri strumenti, invece, i soggetti hanno svolto più rapidamente le operazioni assegnate, impiegando un tempo pari al 54% (107s) del tempo totale utilizzando l'opaque HMD ed al 48% (95s) con il see-through HMD. Il tempo impiegato nello svolgere le operazioni assegnate utilizzando le istruzioni CAI (176s) ha consentito un risparmio pari all'11% del tempo totale. L'impiego della Realtà Aumentata riduce anche il numero di errori commessi. A differenza delle istruzioni *computer-aided*, gli errori risultano pari alla metà, mentre si riducono addirittura ad un quarto rispetto agli errori commessi utilizzando un manuale

cartaceo. Questo avviene perché, rispetto ai mezzi tradizionali, non si verifica un cambio del centro di attenzione dell'utente che, grazie alla visualizzazione delle informazioni in tempo reale e direttamente nella scena inquadrata, non deve muovere la testa o il busto per compiere una ricerca manuale, distraendosi dal compito principale.
L'esperimento di Baird e Bairfield ha comunque evidenziato alcuni limiti delle tecnologie di visualizzazione indossabili per la Realtà Aumentata: molti utenti hanno sottolineato la scarsa ergonomia degli HMD e l'insufficiente contrasto del *see-through* HMD.

3.2. Applicazioni di assistenza all'operatore

I sistemi di supporto all'operatore presenti in letteratura ed analizzati nel presente lavoro sono costituiti generalmente dai seguenti componenti:

- sistema di visione;
- modulo algoritmico che regola la successione (manuale) delle operazioni e gestisce il flusso dati;
- motore di rendering per la Realtà Aumentata.

Il sistema di visione, in queste applicazioni, ha lo scopo di generare la corrispondenza tra la posizione della telecamera nella scena ed il mondo virtuale su cui viene costruito l'aumento di informazioni, per mostrare all'operatore un flusso video quanto più fluido e reale possibile.
L'insieme dei dati che descrive il mondo virtuale raccoglie anche informazioni su: posizioni assolute e relative del macchinario e dei suoi componenti, modelli 3D di oggetti e strumenti per il rendering, gerarchia dei componenti e loro relazioni.
Il cuore di ogni applicazione è il sistema algoritmico che prepara le istruzioni ed i dati necessari per renderizzare le scene di Realtà Aumentata, sulla base degli output provenienti dal sistema di visione e dall'utente stesso che può, in molti sistemi, interagirvi tramite interfaccia grafica.
Infine, il motore di rendering, opportunamente programmato, aggiunge al flusso video gli oggetti virtuali e le istruzioni necessarie all'operatore per guidarlo nei suoi compiti.
Nei sistemi attualmente disponibili l'operatore fruisce informazioni sulle operazioni da compiere ed interagisce con il sistema confermando la loro corretta esecuzione.
Nel seguito del capitolo si analizzeranno alcune applicazioni di assistenza all'operatore, alcune già in commercio, altre in fase di sviluppo. Si confronteranno in modo sommario le caratteristiche hardware e software e si focalizzerà maggiormente l'attenzione sulle analogie e sui limiti dei sistemi in oggetto.

I primi a studiare l'utilizzo dell'AR applicata all'assistenza di un operatore sono stati **Caudell** e **Mizell** all'inizio degli anni novanta, introducendo l'idea di utilizzare un sistema di AR per sostituire i disegni tecnici degli impianti, i manuali e le documentazioni tecniche[14]. Come caso applicativo[8] è stato realizzato un sistema utilizzato nell'assistenza al cablaggio dell'impianto elettrico di aeroplani complessi,

come il Boeing 747, che nella sua ultima versione prevedeva l'utilizzo di un HMD see-through ed un sistema di tracciamento in grado di rilevare dei punti bianchi disegnati su di un pannello di alluminio nero in cui veniva montato il cablaggio elettrico (**Fig.3.2**). Questi punti erano disposti in modo da formare uno speciale marker che permetteva il riconoscimento della posizione e dell'orientazione dell'operatore, in modo da realizzare la parte di AR che consisteva nella visualizzazione di una serie di linee che indicavano la posizione in cui porre i fili del cablaggio elettrico.

Figura 3.2 – Sistema di tracciamento dell'esperimento Boeing.

Feiner e colleghi[17] nel 1993 hanno introdotto il concetto di *Knowledge-based Augmented Reality for Maintenance Assistance* (KARMA) sviluppando un sistema di assistenza alla manutenzione di una stampante laser da ufficio. Questo sistema utilizza un casco di realtà aumentata per la visualizzazione e marker triangolari, prodotti dalla Logitech, per il tracciamento.
I marker sono applicati ai componenti chiave della stampante e permettono al sistema di controllare la loro posizione ed il loro orientamento.
Il sistema, visibile nella **Fig. 3.3**, mostra all'utente come rimuovere il vassoio della carta della stampante laser. La linea rossa più marcata evidenzia il componente tracciato che sta muovendo l'operatore, mentre la linea rossa tratteggiata mostra la destinazione del componente.

Figura 3.3 – Progetto KARMA.

3.2.1. Assemblaggio della serratura di una portiera di automobile

Nel 1998 **Reiners** e colleghi[53] hanno presentato presso la fiera di Hannover un prototipo per l'assistenza guidata dell'assemblaggio di una portiera di un'automobile. Lo studio si inserisce in un progetto di prototipazione virtuale volto ad incrementare la flessibilità del processo di assemblaggio industriale, commissionato da BMW in collaborazione con il Fraunhofer Institut.

Partendo dal modello CAD dell'intera portiera, il sistema ha il compito di mostrare all'addetto il corretto montaggio del blocco di chiusura della portiera.

A causa del ridotto spazio operativo, l'afferraggio e l'inserimento del blocco serratura all'interno dell'assieme sono fasi necessarie per il corretto assemblaggio. Tra le principali criticità dell'operazione vi sono l'inserimento di una piccola leva ed il fissaggio del blocco serratura mediante 3 viti.

Il sistema proposto deve tracciare la portiera per collocare i modelli virtuali nello spazio, mostrare all'utente la corretta presa del blocco serratura ed il movimento da effettuare per inserirlo nella portiera. Sul display viene mostrata la posizione corretta della leva ed una freccia ad indicare la sua posizione spaziale. Infine, vengono mostrati, con alcune animazioni, i fori in cui inserire le viti di bloccaggio.

L'hardware impiegato è costituito da un PC con processore da 180MHz e 128MB di RAM, come visore un HMD Virtual I/O i-glasses[90] al quale è stata collegata una camera Toshiba con un'ottica da 7.5mm e risoluzione 752x582 pixel.

Per tracciare la portiera dell'automobile gli autori hanno scelto di utilizzare un sistema ottico, basato su marker planari posizionati sulla portiera (visibili nelle immagini seguenti). Una voce artificiale guida l'utente attraverso le diverse fasi dell'assemblaggio.

Nelle immagini seguenti si possono osservare alcune fasi della procedura.

Figura 3.4 – Movimento corretto del pezzo.
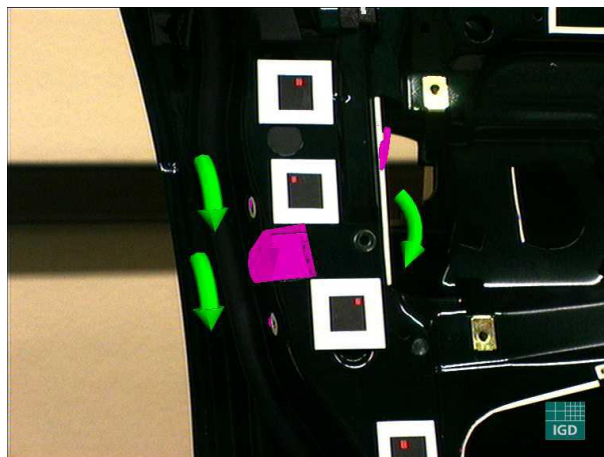
Figura 3.6 – Serraggio dei bulloni.
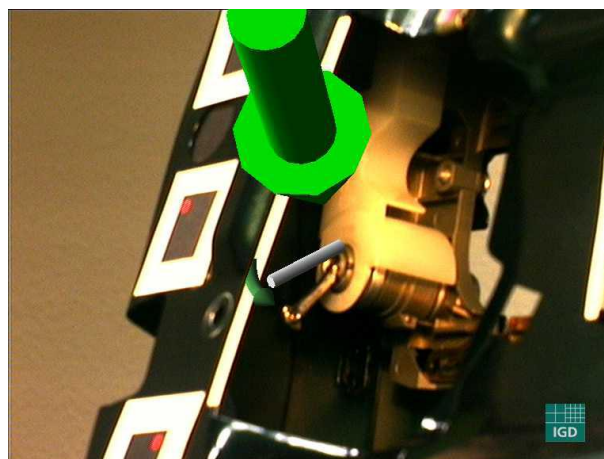
Figura 3.5 – Componente da inserire nella portiera.
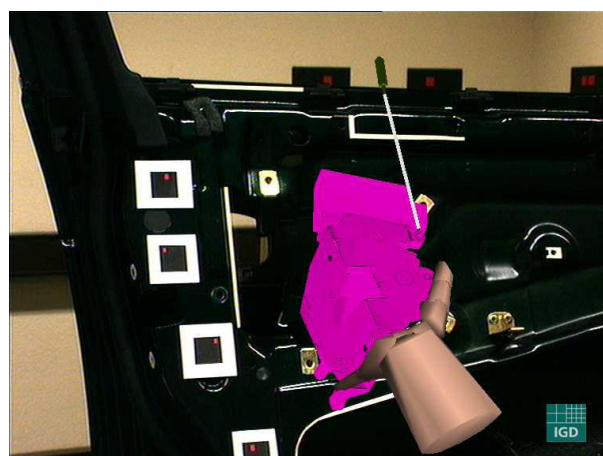
Figura 3.7 – Indicazioni sul corretto afferraggio.

3.2.2. Progetto Metaio

I ricercatori della Metaio[88] hanno sviluppato, nel 2006, un sistema AR[47] che, partendo da un modello CAD di un oggetto presente nella scena, è in grado di calcolare la posizione dell'osservatore, anche in condizioni di non perfetta illuminazione, occlusioni parziali e movimenti rapidi della telecamera, e guidare l'utente attraverso le diverse fasi della manutenzione di un particolare del motore di un'automobile.
L'hardware utilizzato consiste in un notebook (Pentium 1.7 GHz Centrino, 500Mb RAM, scheda grafica NVIDIA FX5650Go), un HMD ed una microcamera[87].
Il vero successo del lavoro proposto, tuttavia, risiede nel sistema di tracciamento markerless sviluppato dalla Metaio. L'algoritmo di tracking è di tipo feature-based ed utilizza un modello CAD per stabilire corrispondenze tra i punti 2D caratteristici dell'oggetto estratto dalla scena reale ed i punti 3D analoghi, estratti dal modello CAD dell'oggetto.
L'intero tracciamento può essere diviso in due fasi principali: una fase di *apprendimento off-line* ed una di *inseguimento on-line*. In **Fig. 3.8** è possibile osservare una schematizzazione di entrambe le fasi.
L'obiettivo della fase di apprendimento è la creazione di una serie di **key frames**, immagini dell'oggetto che dovrà essere tracciato in real-time con informazioni aggiuntive sulle corrispondenze 2D-3D, da cui sarà ricostruita la camera pose.
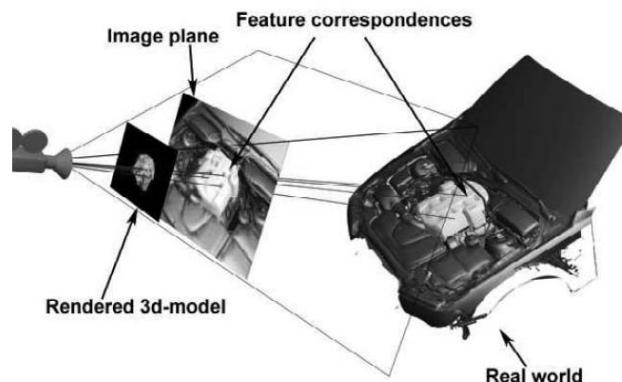
Figura 3.8 – Estrazione e matching delle feature.

Il salvataggio dei key frames impiega un sistema di tracking ausiliare, basato su marker planari di dimensione e posizione nota, che saranno poi rimossi durante il funzionamento on-line del sistema.
Il set di key frames dovrebbe essere il più ampio possibile, per coprire l'intero spazio operativo e consentire la re-inizializzazione del sistema da qualsiasi angolazione e posizione di lavoro dell'utente.
La fase di inseguimento on-line si articola nelle operazioni di inizializzazione e tracciamento.

L'inizializzazione prevede che, per calcolare la camera pose iniziale, ogni fotogramma acquisito dalla webcam sia confrontato con l'intero set di key frames. Quando le corrispondenze tra features superano un valore di soglia, si assume che la camera pose reale coincida con quella del key frame associato alle corrispondenze estratte. A partire dalla posizione iniziale appena calcolata, il sistema impiega l'algoritmo KLT[43] per eseguire il tracciamento ed il matching delle feature estratte tra immagini successive nel flusso video reale.
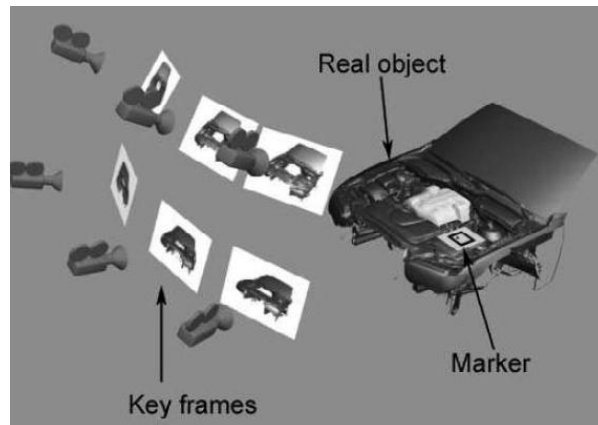
Figura 3.9 – Acquisizione dei key frames.

Il sistema è stato testato sia in ambiente reale che virtuale.
Per quanto concerne l'ambiente virtuale sono state create più sequenze video di 100 frames facendo un rendering del modello CAD dell'automobile e variando sia la complessità delle operazioni che le caratteristiche ambientali (occlusioni ed illuminazione).
Variando la distanza da 500mm a 1600mm, l'RMS dell'errore sulla posizione della telecamera, in tutte le sequenze, è di circa 5mm per la traslazione e 0.8° per la rotazione.
Per la valutazione nell'ambiente reale non vengono forniti dati sulla precisione ma viene specificato che tutte le operazioni di manutenzione sono andate a buon fine, in particolare anche con la presenza di occlusioni dovute agli arti dell'operatore, movimenti rapidi e forti variazioni di distanza tra camera e area di lavoro. In **Fig. 3.10** si possono osservare alcuni fotogrammi di una delle sequenze di test.

I **limiti** del sistema appena descritto appaiono principalmente due e sono relativi alla scelta dell'utilizzo dei key frames. La forte dipendenza dei key frames dalle condizioni di illuminazione pone il vincolo alle forti variazioni di illuminazione tra la fase di acquisizione dei key frames e l'applicazione. Questo vuol dire che tra le due fasi non deve intercorrere un lungo lasso di tempo, oppure che l'intero sistema dovrebbe operare in condizioni controllate di luce ambientale.

In secondo luogo, per una corretta inizializzazione della camera pose è necessario un cospicuo numero di key frame, o, in alternativa, che l'utente parta da una posizione nota a priori.
La ragione di questo comportamento è legata al raggio di tracciamento dell'algoritmo KLT, che utilizza un'area di ricerca ottimizzata per il calcolo delle corrispondenze.
Per migliorare le prestazioni della procedura di inizializzazione gli autori hanno introdotto una procedura di training per eliminare le features inaffidabili durante la fase di apprendimento: dopo l'acquisizione di un key frame viene eseguito un breve inseguimento dello stesso per rimuovere possibili falsi positivi.
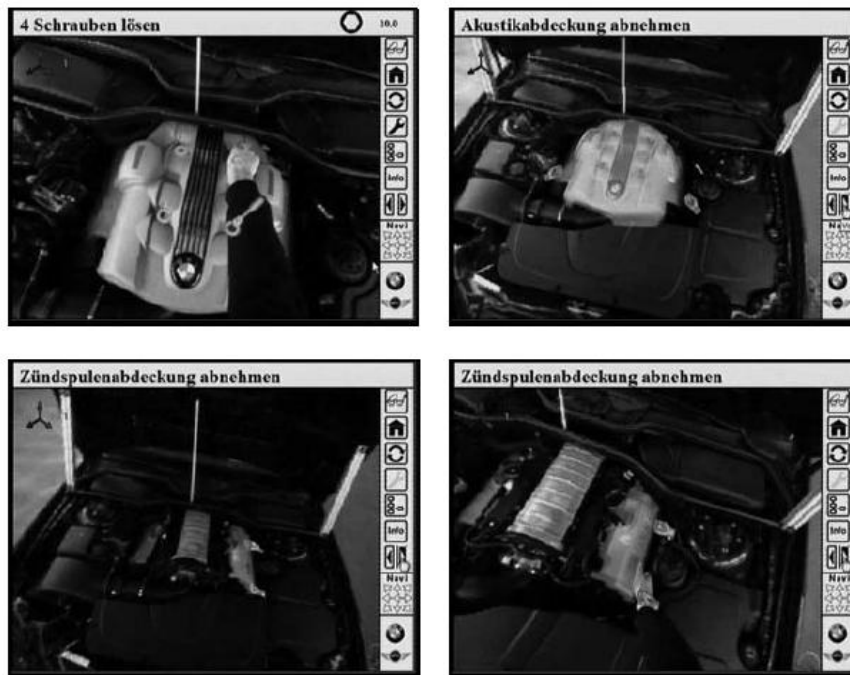
Figura 3.10 – Sequenza operativa dell'applicazione METAIO.

3.2.3. Progetto ARMAR

Il progetto ARMAR (Augmented Reality for Maintenance and Repair)[26] è stato realizzato nel 2009 presso la Columbia University (USA) con lo scopo di valutare l'effetto della Realtà Aumentate su produttività, precisione e sicurezza del personale addetto alla manutenzione.
Il lavoro si differenzia dagli esempi precedenti nella letteratura del settore per due peculiari caratteristiche della ricerca. In primo luogo, il sistema sviluppato è stato testato su un vasto campione di tecnici manutentori esperti, in un'ampia casistica di applicazioni specifichee utilizzando differenti condizioni di confronto. Tra le applicazioni si evidenzia la manutenzione di una torretta di un carro armato, realizzata con il supporto della Marina degli Stati Uniti[75].

Un altro aspetto su cui si è posta particolare attenzione è stato quello relativo all’interfaccia utente. Diverse metodologie di progetto sono state seguite per studiare il movimento della testa dell’operatore con l’obiettivo di sviluppare un’interfaccia che consentisse di indirizzare l’attenzione dell’operatore e ridurre i movimenti superflui. In particolare, sono state oggetto di studio 5 tipologie di contenuti con cui arricchire l’informazione da fornire all’utente:

- Frecce 2D/3D convergenti per catturare l’attenzione dell’utente e guidarne le rotazioni della testa;
- Caratteri e dimensioni delle istruzioni in forma testuale con cui descrivere le operazioni e le precauzioni da adottare;
- Etichette per indicare oggetti noti nella scena;
- Modelli 3D di utensili da visualizzare nelle posizioni in cui dovranno essere utilizzati gli utensili reali, con animazioni sul loro specifico utilizzo;
- Ingrandimento dei particolari su cui compiere un’operazione.

.

Figura 3.11 – Progetto ARMAR. Esecuzione di un'operazione di manutenzione.

Il sistema è stato testato con un HMD della Headplay con una risoluzione 800x600 pixel sul quale sono state montate due telecamera Point Grey Firey MV con una risoluzione 640x480 pixel collegate al PC tramite un bus IEEE 1394a. L'applicazione viene fatta girare su di un PC Windows XP con una scheda grafica NVIDIA Quadro 4500.

Il tracciamento è stato eseguito grazie al sistema NaturalPoint dell’OptiTrack [89].

La torretta dell’applicazione presenta un volume di lavoro molto limitato e ricco di parti strutturali, che occludono la scena, creano molti punti morti. È stato necessario impiegare circa 10 telecamere per ottenere un tracking robusto che coprisse l’intera torretta.

In un primo momento è stato scelto un sistema di marker passivi infrarossi, si è poi optato per l’impiego di marker attivi a causa di riflessi dovuti al materiale metallico della torretta, collocando alcuni LED sull'HMD.

Figura 3.12 – Sequenza di localizzazione di un componente.

I risultati dei test hanno confermato che l'impiego della Realtà Aumentata consente un incremento della produttività. Il tempo medio di completamento delle attività, rispetto al tempo di normale esecuzione, è risultato nettamente inferiore. Inoltre, la contemporanea presenza di informazioni ed oggetto della manutenzione all'interno della stessa area inquadrata consente un notevole risparmio di tempo, riducendo distrazioni dell'operatore e movimenti non sempre confortevoli, specialmente in volumi di lavoro ristretti.

3.2.4. Progetti ARVIKA ed ARTESAS

Fondato alla fine degli anni '90, ARVIKA[20] rappresenta uno dei maggiori progetti di ricerca nel campo dell'AR voltata a trattare la manutenzione e l'assemblaggio. Il Ministero Tedesco dell'Istruzione e della Ricerca ha finanziato, nell'arco degli anni 1999-2003, lo sviluppo di numerose applicazioni nel settore automotive, aerospace, power processing e nel settore delle macchine utensili.
Il progetto ARVIKA coinvolge numerosi partners provenienti da differenti contesti, quali quello industriale (Volkswagen, BMW, Airbus, Siemens), della ricerca (Fraunhofer, ZVGD) e universitario (Università di Munich e Aachen).
I principali temi affrontati da ARVIKA sono legati alla sperimentazione della Realtà Aumentata nella produzione e manutenzione di un prodotto o di una macchina. On-line[83] è possibile reperire i risultati ottenuti con le varie ditte partner del progetto.
Il progetto ARTESAS (Advanced Augmented Reality Technologies for Industrial Service Applications)[24] è nato nel 2006 a partire dai risultati ottenuti dal progetto ARVIKA, ed intende valutare le tecnologie basate sull'AR per applicazioni in ambienti industriali.
Il progetto si focalizza principalmente sui seguenti temi[82]:

- Procedure di tracciamento markerless in ambienti industriali;
- Metodologie di interazione con i dispositivi per la Realtà Aumentata;
- Implementazione e valutazione in applicazioni industriali.

3.2.5. Altri Progetti e Applicazioni Commerciali

Il forte interesse dell'industria per le applicazioni della realtà aumentata ha permesso, negli ultimi anni, la nascita ed il finanziamento di numerosi progetti, quali quelli citati in precedenza, volti a studiare e analizzare l'impatto delle nuove tecnologie nelle fasi industriali di progetto e produzione.
Tra i numerosi esempi, citiamo anche:

- progetto PLAMOS[78]

Progettato e sviluppato dal VTT Technical Research Centre of Finland, il sistema si propone l'obiettivo di assistere gli ingegneri di processo nella gestione e manutenzione degli impianti di potenza. La ricerca si è concentrata principalmente sull'interazione con l'utente. L'obiettivo principale del sistema è quello di coordinare la varietà di dati disponibili in un unico flusso omogeneo di informazioni per l'utente, intervenendo in tempo reale per segnalare gli interventi ordinari e straordinari sui processi da controllare.

- progetto MANUVAR [77]

ManuVAR è un progetto europeo che coinvolge numerosi partner (aziende private ed enti di ricerca) provenienti da 8 nazioni. Attraverso l'impiego di tecnologie per la realtà virtuale ed aumentata, il progetto vuole fornire strumenti innovativi per migliorare la qualità del lavoro manuale industriale, arricchendone il valore e l'alto contenuto di conoscenze tecniche per aumentare la competitività delle aziende europee.

Intervenendo sull'intero ciclo produttivo, avvalendosi di diversi supporti tecnologici, quali strumenti di visione e sensori aptici, il progetto vuole migliorare l'ergonomia, la sicurezza, l'assistenza e la formazione in ambito industriale.

- progetto AugAsse[92]

Augmented Assembly è un progetto di ricerca sviluppato anch'esso dal VTT. La realtà aumentata è utilizzata per assistere gli addetti all'assemblaggio di componentistica, visualizzando istruzioni testuali e modelli 3D su diversi supporti. Si osservano il sistema di tracciamento marker-based e l'impiego di sensori che forniscono un feedback sulle operazioni eseguite.
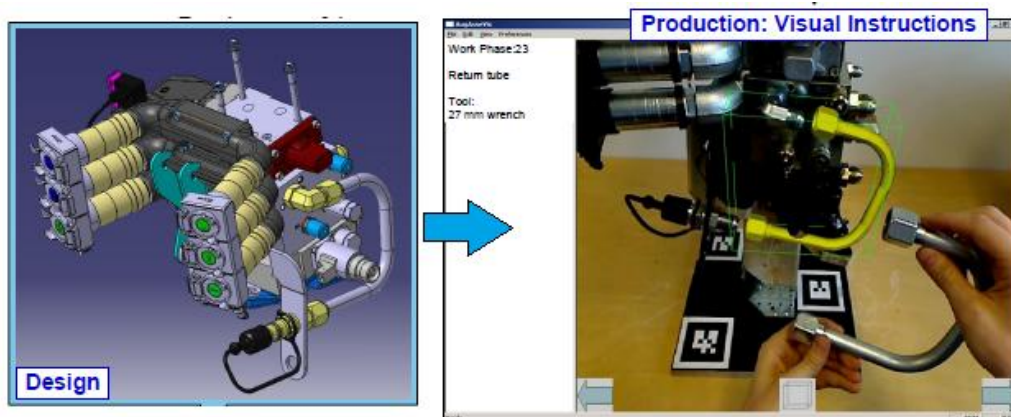
Figura 3.13 – Progetto AugAsse.

- progetto RAMAR[86];

Il progetto è stato condotto dall'università di Pisa e prevedeva la realizzazione di una prima fase di un percorso più ampio, volto a mettere a punto una metodologia innovativa e la necessaria dotazione strumentale, sia hardware che software, per lo svolgimento di attività di manutenzione, assistenza e formazione remota da utilizzare in specifici contesti industriali.
Tale fase era rivolta a realizzare e testare un prototipo di sistema di realtà aumentata, per analizzare delle problematiche connesse al suo impiego nei settori target della ricerca.

Si registrano, inoltre, alcune interessanti applicazioni commerciali, proposte da aziende italiane, quali ad esempio:

- R.E.A.L.[79] è un **sistema portatile di assistenza remota** che permette al personale tecnico di ricevere **supporto audio e video da esperti a distanza** attraverso uno strumento di visualizzazione delle informazioni che si può indossare. REAL permette assistenza tramite realtà aumentata in real-time per fini formativi, seguendo gli utenti a distanza passo dopo passo in ogni fase di un progetto: **assistenza, manutenzione, risoluzione dei problemi.**

- Augmented XP[85] è un prodotto sviluppato per l’assistenza dei tecnici manutentori nell’industria delle telecomunicazioni. Attraverso un sistema composto da HMD e webcam, il tecnico sarà supportato nel riconoscimento del component su cui operare e potrà ottenere informazioni aggiuntive in tempo reale sulle operazioni da compiere.

Figura 3.14 – Augmented XP (JoinPAD).

3.3. Osservazioni conclusive

Nei sistemi analizzati ed in altri presenti in letteratura si è riscontrato un limite nel controllo delle azioni dell'operatore. Spesso, infatti, è richiesto un input dell'operatore per avanzare al passo successivo dell’esecuzione, senza la possibilità di controllare l’effettiva correttezza del lavoro svolto. L’interazione dell’utente con il sistema, in questo modo, si riduce all’esecuzione di un’azione visualizzata e al confermarne l’avvenuto completamento al sistema. Un'ulteriore criticità riscontrata nei sistemi di assistenza tramite realtà aumentata sviluppati fino a questo momento è l'utilizzo di marker fiduciali per valutare la posizione della macchina o di una parte di essa.
Nell'architettura progettata e sviluppata nel presente lavoro di tesi, di cui si parlerà ampiamente nei capitoli successivi, cambia il ruolo dell’utente che, da utilizzatore di informazioni, diventa generatore delle stesse, con l’obiettivo di poter verificare in automatico la bontà delle operazioni compiute.
Integrando tecniche di computer vision per il riconoscimento ed il tracciamento di oggetti in real-time (eventualmente utilizzando anche sensori e strumenti più specifici) è possibile analizzare in tempo reale cosa l’operatore sta facendo e come sta eseguendo le operazioni, implementando una nuova funzionalità per sistemi di assistenza all'operatore analoghi a quelli analizzati.

Il sistema proposto in questo lavoro di tesi contiene nuovi moduli e funzionalità innovative:

- un sistema di visione per identificare ed inseguire oggetti in movimento;
- un sistema di visione per il tracciamento dello stato della macchina;

- un sistema di supporto alle decisioni in grado di valutare se un'operazione è stata correttamente eseguita da parte dell'operatore e di adattare il flusso logico della procedura di conseguenza;
- un modulo di apprendimento e selezione di oggetti a supporto dell'interazione uomo – macchina.

Il sistema in oggetto si propone di superare la criticità dei marker fiduciali utilizzando come "marker" features naturali presenti all'interno dell'ambiente, evitando quindi un passaggio critico per l'operatore ovvero il corretto posizionamento dei marker all'interno dell'ambiente.

Infine, un altro aspetto su cui si è posta l'attenzione è la fase di istruzione del controllo.
Nonostante il sistema proposto sia stato progettato e sviluppato per essere utilizzato principalmente durante la fase operativa di esecuzione e controllo delle operazioni di manutenzione, è stato ritenuto necessario implementare un ambiente di editing, in cui un tecnico istruttore potesse configurare il sistema sulle operazioni da monitorare e sugli eventuali errori da gestire, in maniera grafica e schematica. L'analisi e lo sviluppo della fase di training off-line si inserisce nello studio delle metodologie e dei processi che cercano di migliorare la qualità dell'interazione uomo-macchina in sistemi complessi ed automatizzati, quale quello in esame.

Capitolo 4
Progettazione e Sviluppo del sistema

In questo capitolo saranno analizzate le fasi di progettazione e sviluppo del sistema proposto, si descriveranno, inoltre, l'architettura del sistema e gli algoritmi scelti per la sua realizzazione.
Condotto con la finalità di assistere un utente in fase di training, nelle operazioni di assemblaggio, attrezzaggio o manutenzione di una macchina, il lavoro svolto si inserisce fra gli strumenti a supporto dell'interazione uomo-macchina tramite realtà aumentata. Come già precisato nel capitolo 3 la differenza con molti dei lavori presenti in bibliografia consiste nello sviluppo delle potenzialità offerte dal sistema di visione per la verifica e la correzione, in tempo reale, dell'esecuzione delle operazioni.
Il rendering di modelli 3D e l'inserimento nel flusso video di scritte informative, con tecniche di realtà aumentata, migliorano la qualità dell'interazione, informando l'utente sulle operazioni da svolgere ed avvisandolo in caso di errata esecuzione.
Poiché la progettazione e lo sviluppo del sistema presentato in questa Tesi di Laurea non sono state fasi nettamente distinte del lavoro, ma sono state condotte in modo concatenato e ricorsivo nell'applicazione di un metodo progettuale ben strutturato, si è scelto di presentarle, nel seguito del capitolo, contestualmente per ogni parte del sistema in oggetto, in modo da sottolinearne il forte legame concettuale e pratico al tempo stesso.

4.1. Il sistema completo

Nel presente lavoro di tesi è stato realizzato un sistema di controllo e supporto alle operazioni dell'utente in grado di verificare il flusso logico della sequenza di operazioni da compiere, potenziando il concetto di sistema manualistico e di assistenza con la Realtà Aumentata.
Il sistema è in grado di restituire all'operatore, tramite l'impiego in tempo reale della Realtà Aumentata, una retroazione sul passo eseguito in modo da intervenire su eventuali errori ed imprecisioni o avanzare, senza necessità di ulteriori comandi, fino alla completa esecuzione della procedura da svolgere.
La realizzazione di un sistema complesso, quale quello in esame, ha condotto allo sviluppo di un modulo di visione in grado di riconoscere features naturali presenti all'interno di un ambiente di lavoro non strutturato, allo scopo di tracciare la posizione della telecamera nello spazio e garantire un sistema di riferimento per le tecniche di Realtà Aumentata impiegate. Sono stati, inoltre, studiati ed integrati nel sistema algoritmi di visione artificiale in grado di identificare ed inseguire oggetti tridimensionali, riconoscere particolari forme o colori in un'immagine ed analizzare lo stato dell'ambiente di lavoro per estrarre informazioni utili al sistema.
Per evidenziare in tempo reale le istruzioni su passaggi da effettuare e componenti da manipolare è stata realizzata una classe di funzioni per la Realtà Aumentata che, a partire dalle informazioni sul tracciamento dei marker naturali presenti nell'ambiente, visualizza scritte e modelli tridimensionali sul display utilizzato dall'utente.
È stato progettato e sviluppato un modulo di controllo che gestisca le diverse funzionalità di cui sopra, valuti la correttezza delle operazioni eseguite dall'operatore e ne supporti le decisioni. Il modulo di controllo dovrà, infatti, coordinare e codificare lo scambio di dati tra macchina ed utente e, sulla base dell'output restituito dagli algoritmi di computer vision, adattare il flusso logico del processo sulle azioni intraprese dall'utente, mostrando un opportuno feedback all'operatore.
L'esecuzione in modo automatico ed in tempo reale dell'intero sistema è guidata da un insieme codificato di istruzioni, regole di funzionamento e di informazioni che descrivono le operazioni da compiere, i dati da analizzare e le azioni da intraprendere, seguendo una casistica, opportunamente predefinita, di condizioni possibili alle quali riferirsi.
Così come una procedura completa di manutenzione si articola in una sequenza ordinata di azioni codificate da eseguire, anche il sistema realizzato in questa Tesi esegue l'insieme codificato di istruzioni di cui sopra.
In analogia con le operazioni da monitorare, sono state parametrizzate alcune azioni elementari che il sistema dovrà eseguire durante la fase operativa.

Si definiscono a questo punto due termini, che saranno ricorrenti nel seguito del capitolo, il cui significato è di fondamentale importanza per il resto della dissertazione:

- Si definisce **passo operativo** l'operazione che viene eseguita dal sistema per esercitare il controllo sulla singola azione dell'utente;
- Si definisce **procedura** l'insieme dei passi operativi. Più discorsivamente, la procedura è l'intera sequenza di operazioni codificate e di informazioni che il sistema dovrà gestire mentre l'utente svolge il proprio compito.

Il passo operativo è descritto e rappresentato da una struttura dati che contiene l'informazione necessaria all'esecuzione della singola operazione logica.
Più in dettaglio, ogni passo operativo contiene i messaggi di testo da visualizzare all'utente, gli eventuali modelli 3D da visualizzare in tempo reale a supporto dell'utente, i riferimenti necessari all'esecuzione in automatico degli algoritmi di visione selezionati per l'operazione corrente e tutti i valori ammissibili per le variabili da monitorare, a cui saranno associate le possibili configurazioni di completamento del passo operativo all'interno della procedura.
Il sistema proposto nel presente lavoro si articola, in due fasi distinte a cui corrispondono due indipendenti modalità di esecuzione:

- la redazione ed il salvataggio della procedura, che il sistema seguirà per assolvere uno specifico compito. Questa fase si definisce ***fase istruttore***;
- il caricamento e l'esecuzione della procedura, che è l'assistenza guidata vera e propria al personale tecnico addetto al suo svolgimento. Questa fase si definisce ***fase operatore***.

Per entrambe le fasi di esecuzione del sistema è stata sviluppata un'applicazione software specifica: la fase istruttore è eseguita dall'applicazione ***istruttore.exe*** , la fase operatore è eseguita dall'applicazione ***operatore.exe***.
L'utente della fase istruttore, chiamato *istruttore* del sistema, ha il compito di eseguire il **training off-line** del sistema. Dovrà creare la procedura di esecuzione delle operazioni, sfruttando la struttura modulare per il controllo delle operazioni e l'insieme di funzioni logiche primitive di Computer Vision e strumenti per il loro utilizzo presentate in questa tesi, e fornire al sistema gli input da ricercare e gli output da restituire all'utente finale. L'istruzione del sistema avviene definendo la sequenza dei singoli passi operativi, definita in precedenza e nel seguito con maggior dettaglio, che compongono la procedura da eseguire e verificare (**Fig. 4.1**).
Per ogni passo operativo l'istruttore deve descrivere gli stati iniziale e finale, rispettivamente la condizione di inizio operazione e quella di corretto completamento. Per ogni passo operativo, inoltre, è possibile individuare e segnalare al sistema alcune verifiche aggiuntive, associando ad esse le relative condizioni di incompleta, o errata, esecuzione. Per la compilazione dell'insieme contente le regole della procedura di funzionamento è stato sviluppato un software (come già accennato, istruttore.exe) che, grazie ad un'interfaccia grafica, guida l'utente nella creazione e nel salvataggio del file relativo alla procedura in un database.
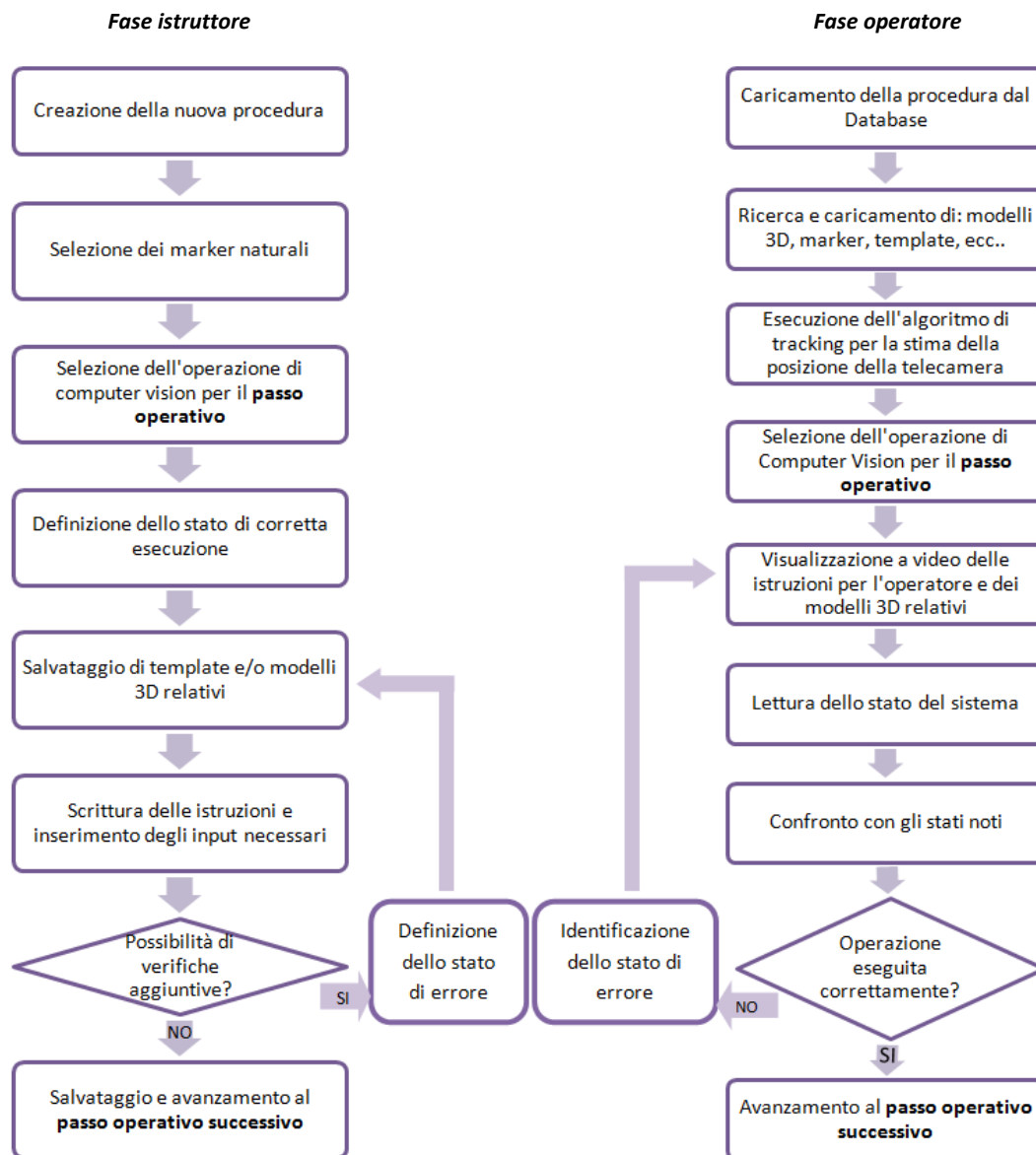
Figura 4.1 – Struttura logica di funzionamento del sistema nelle fasi istruttore (a sinistra) e operatore(a destra) nella creazione e nell'esecuzione del singolo passo operativo della procedura.

Il compito dell'utente finale (fase operatore, applicazione operatore.exe), che espleta materialmente l'attività manutentiva (**Fig. 4.1**), è quello di selezione e caricamento della procedura specifica, necessaria allo svolgimento ed all'assistenza del proprio compito. L'applicazione, seguendo quanto definito nella procedura caricata, utilizza tutte le informazioni in essa contenute per tracciare quanto operato dall'utente e visualizzare le relative informazioni e notifiche tramite l'interfaccia AR.

Il comportamento delle due fasi (istruttore e operatore) appena descritte è rappresentato nei due algoritmi in **Fig. 4.1**.
Un sistema così realizzato può essere impiegato nella formazione del personale tecnico, per il supporto allo svolgimento di operazioni critiche di assemblaggio e per la manutenzione di macchinari. Dato l'ampio spettro di applicazioni, si è scelto di sviluppare tutte le funzionalità del sistema in modo da poter essere utilizzate con flessibilità. È possibile, infatti, modificare i parametri di ingresso o l'ordine delle operazioni, oppure la tipologia della condizione da ricercare in modo di programmare il sistema per l'esecuzione in un contesto diverso, con altre esigenze operative.
Per gestire in modo efficiente le diverse funzionalità del sistema e la complessità dello scambio di dati differenti fra loro è stata progettata una struttura modulare che raccoglie tutte le funzioni sviluppate in aree comuni per obiettivi e finalità.
La gestione della procedura è stata implementata con un database che viene opportunamente riempito nella *fase istruttore* seguendo una procedura guidata tramite l'apposito software.
In **Fig. 4.2** è schematizzata l'architettura del sistema. Se ne possono immediatamente osservare la struttura modulare e l'interazione reciproca fra i vari moduli.
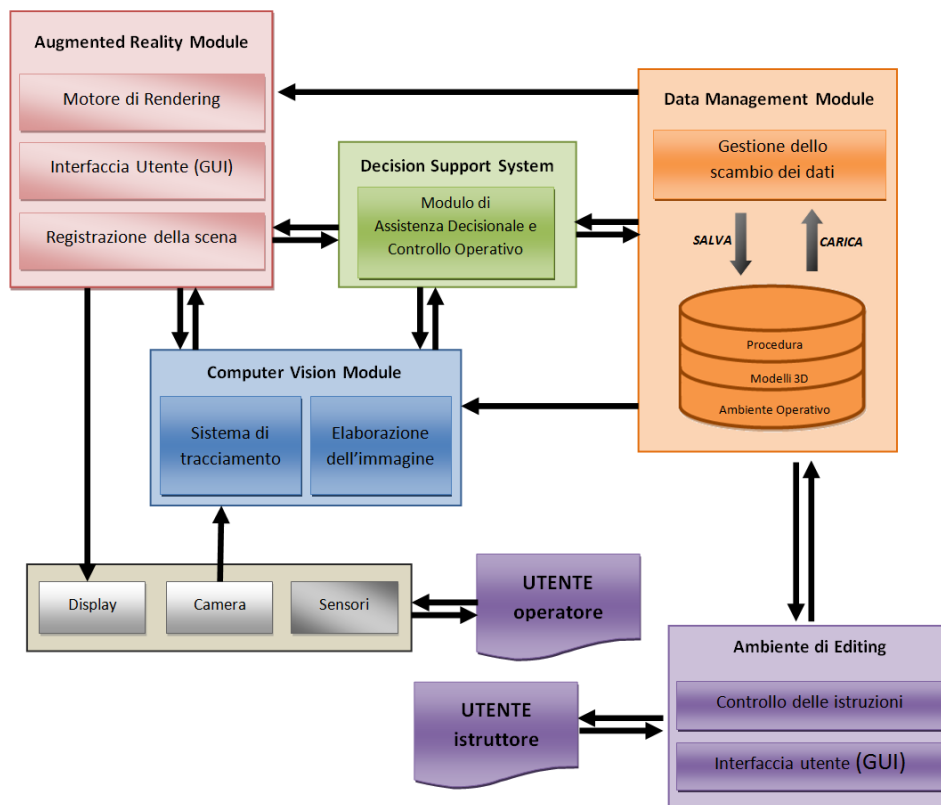
Figura 4.2 – Architettura del sistema completo.

Nel seguito del capitolo si analizzeranno nel dettaglio la struttura, lo sviluppo e le funzioni di ciascuno dei seguenti moduli, che sono i blocchi funzionali progettati ed implementati per la fase operatore (applicazione operatore.exe):

- **Computer Vision Module** (CVM) interagisce direttamente con il sistema di visione per elaborare le immagini acquisite dalla telecamera ed applicare gli algoritmi di computer vision necessari al controllo sull'ambiente in cui si sta operando. Il modulo di visione implementa il sistema di tracciamento markerless, realizzato nel presente lavoro di tesi, che permette di individuare nella scena osservata più marker naturali contemporaneamente, consentendo un agganciamento periodico alle features naturali presenti nella memoria del sistema. Sono state sviluppate, inoltre, specifiche funzioni di computer vision per individuare oggetti, utilizzando algoritmi di pattern matching e blob detection[1].
- **Augmented Reality Module** (ARM) è il modulo che gestisce le tecniche di realtà aumentata, con l'ausilio delle quali restituisce all'utente feedback visivi sulla procedura che sta eseguendo, per informarlo su eventuali errori commessi o sulla correttezza dell'esecuzione. Implementa le funzioni per il rendering di scritte e modelli 3D, a partire dal caricamento dall'hard disk del computer dei file dei modelli fino alla loro collocazione nello spazio per l'applicazione di Realtà Aumentata, utilizzando i dati del tracciamento della telecamera.
- **Decision Support System** (DSS) questo modulo ha il compito di interrogare gli altri moduli e valutare le informazioni da questi ricevute, al fine di prendere decisioni sullo svolgimento dell'attività dell'operatore. Rappresenta il cuore algoritmico dell'intero sistema perché fulcro del sistema di assistenza decisionale per l'utente. Il sistema di assistenza alle decisioni attinge le informazioni dal database e, attraverso l'impiego di variabili simboliche codificate, procede fra le operazioni da effettuare in modo automatico e consequenziale.
- **Data Management Controller** (DMC) è il modulo di gestione dello scambio dati fra i moduli del sistema. Al suo interno è presente il database che conterrà tutte le informazioni necessarie al buon esito dell'applicazione ed al suo completo funzionamento in automatico. In particolare vi sarà immagazzinata la procedura di funzionamento, che stabilisce le regole di controllo sulle operazioni che l'utente dovrà eseguire.
- **Editing Environment** (EE) è il modulo relativo all'ambiente di creazione della procedura (fase istruttore, applicazione istruttore.exe). Il modulo non è coinvolto durante la fase operatore ma è ovviamente necessario per gestire la configurazione dell'intero sistema prima del suo impiego.

---

[1] In Computer Vision le tecniche di blob detection hanno lo scopo di individuare all'interno di un'immagine delle aree che si distinguono per particolari caratteristiche, quali colore o luminosità, rispetto al contesto dell'intera immagine.

4.2. Computer Vision Module

Il modulo di Computer Vision (CVM) utilizza il sistema di visione per estrarre informazioni dall'ambiente osservato. Il flusso dei dati raccolti sarà inviato agli altri moduli per essere interpretato e trasformato in informazioni per l'utente. Il CVM raccoglie ed implementa tutti gli algoritmi di visione artificiale necessari al funzionamento in automatico ed in tempo reale del sistema, per il controllo dell'esecuzione della procedura ed il tracciamento della posizione della telecamera in un sistema di riferimento assoluto nello spazio.
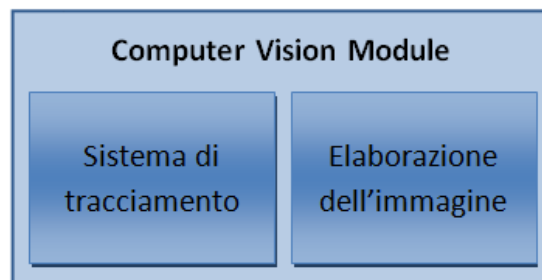
Figura 4.3 – Modulo di Computer Vision.

Il modulo di computer vision si presenta idealmente diviso in due parti, in relazione alle principali finalità degli algoritmi di visione artificiale. Le funzioni relative al Sistema di Tracciamento (o **Camera Pose Tracking)** calcolano e trasmettono agli altri moduli le rotazioni e le traslazioni del sistema di visione nello spazio. Le funzioni di Elaborazione dell'Immagine (o **Image Processing)**, invece, rappresentano gli strumenti che il sistema di supporto decisionale interattivo (DSS) utilizza per analizzare lo stato del sistema ed intervenire in modo coerente, laddove necessario.
Sulla teoria degli algoritmi per il tracking si è discusso ampiamente nel primo capitolo, nel seguito si descriverà il loro sviluppo, oggetto del presente lavoro di tesi. Per quanto riguarda le altre funzionalità di computer vision sviluppate è bene affrontare la seguente premessa concettuale per comprendere le ragioni che hanno orientato la scelta di alcune particolari funzioni.
La maggior parte delle operazioni di montaggio può essere ridotta ad una sequenza di azioni elementari che, come sancito dai principi del *design for assembly*[62] e da Boothroyd[11], è rappresentata dalle seguenti operazioni (**Fig. 4.4**):

- presa del corretto componente;
- allineamento del pezzo;
- posizionamento del componente;
- inserimento temporaneo o definitivo del componente.

Analogamente a quanto avviene per la programmazione di un robot[45], si è cercato di creare un parallelismo tra le operazioni elementari che un tecnico compie durante l'assemblaggio o la manutenzione di un particolare meccanico e le operazioni di computer vision necessarie al sistema per esercitare il controllo sulle suddette azioni[67].
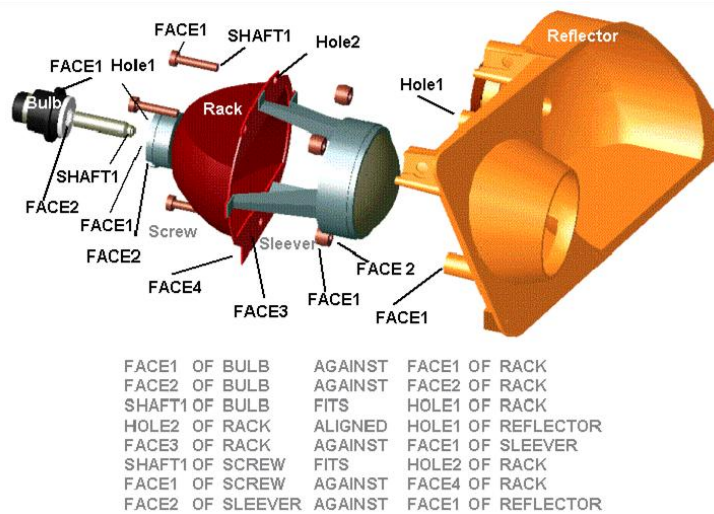
Figura 4.4 – Esempio di una sequenza di operazioni di assemblaggio.

Sono state individuate alcune funzioni primitive da associare alle operazioni elementari secondo il seguente schema:

Operazioni logiche	Funzioni Primitive di Computer Vision
• **Selezione di un oggetto nella scena.**	• Salvare un template. • Salvare un marker.
• **Riconoscimento di un oggetto noto a priori.**	• Inseguire un template. • Tracciare un marker.
• **Riconoscimento di un oggetto non noto, ma di cui si conoscono la geometria (*) o la posizione(**) in cui dovrebbe essere.**	• Riconoscere una forma. (*) • Identificare un blob. (**) • Verificare la presenza di un blob in una ROI (Region of Interest).
• **Verificare il corretto posizionamento di un pezzo in una regione (ROI) della scena.**	• Verificare la presenza di un template/marker in una ROI.
• **Inserimento/rimozione di un pezzo** • **Contatto relativo fra due oggetti** • **Allineamento fra due pezzi**	• Identificare un blob all'interno di una ROI. • Confrontare due immagini della stessa scena.

Tabella 4.1 – Tabella delle operazioni primitive.

Le operazioni primitive identificate e descritte in **tabella 4.1** rappresentano la base del sistema proposto per interagire con l'ambiente ed operare il controllo sulle azioni dell'utente. È possibile, infatti, creare sequenze diverse con le stesse operazioni per assolvere obiettivi diversi. Questa scomposizione in operazioni di controllo elementari

si rivela strategica per poter individuare e risolvere il maggior numero di situazioni possibili che l'utente si troverà a fronteggiare.
Il modulo di computer vision è stato, quindi, sviluppato seguendo la divisione concettuale descritta in precedenza.

4.2.1. Camera Pose Tracking

Utilizzando l'algoritmo SURF, descritto nel capitolo 1, per l'estrazione ed il riconoscimento delle features naturali presenti nella scena inquadrata dalla telecamere ed altri algoritmi per il calcolo delle corrispondenze, quali ad esempio FLANN e RANSAC (vedi cap. 1 e [18][50]), è stato scritto un modulo per il tracking di oggetti texturizzati noti presenti nella scena acquisita dalla webcam.
Una volta riconosciuti gli oggetti, il modulo ne restituisce la posizione nello spazio, utilizzando la matrice di trasformazione prospettica ed i parametri intrinseci, calcolati con la calibrazione della telecamera. L'implementazione dell'algoritmo SURF è contenuta all'interno della libreria OpenCV (Appendice B) e si articola in due fasi: l'estrazione di Keypoints dalla scena ed il calcolo di un descrittore per ogni feature identificata. Queste informazioni vengono salvate in due variabili (i vettori contenenti gli elementi, definiti dall'algoritmo, detti *Keypoints* e *Descriptors*) per un successivo confronto delle features presenti all'interno del frame acquisito dalla telecamera. Le immagini dei marker naturali da ricercare all'interno della scena vengono caricate dalla memoria del computer. Keypoints e Descriptors di ogni marker vengono inseriti in un array formato da tante righe quante sono le immagini caricate.
Dopo aver costruito il set di dati relativi ai marker, si procede alla ricerca degli stessi nel video real-time acquisito dalla webcam, seguendo la logica descritta nel diagramma di flusso in **Fig. 4.7** ed esposta nel seguito.
Il frame acquisito viene prima convertito in scala di grigi e successivamente filtrato dalle brusche transizioni e dal rumore (componenti ad alta frequenza) tramite l'applicazione di un filtro di smoothing di tipo Gaussiano.
Dal frame, così elaborato, si estraggono i punti di interesse ed i descrittori e, tramite l'applicazione dell'algoritmo FLANN (anch'esso implementato all'interno della libreria OpenCV attraverso la classe `FlannBasedMatcher`), si valutano le corrispondenze tra i descrittori dell'immagine acquisita e quelli del pattern precedentemente salvati.
Affinché la stima della camera pose possa essere considerata attendibile dal sistema e non generi falsi positivi è necessario un numero minimo di corrispondenze, stabilito dalla tipologia di operatore utilizzato. In questo caso si richiede che le corrispondenze rilevate con i marker in memoria siano almeno su quattro keypoints.
Per poter operare il confronto con diversi pattern è stato creato un indice che identifica la posizione di ogni immagine nell'array del data-set dei marker. Il programma esegue il tracking di un marker alla volta, se non trova più il marker precedentemente individuato nella scena, incrementa l'indice e scorre tra le righe dell'array per cercarne un altro.
Per ogni pattern riconosciuto si applica il metodo `findHomography` di OpenCV per ricavare la matrice di omografia. Si utilizza il metodo iterativo del *Random Sample*

Consensus[18], o RANSAC, per la stima dei parametri di un modello matematico a partire da un insieme di dati contenente anche outliers.
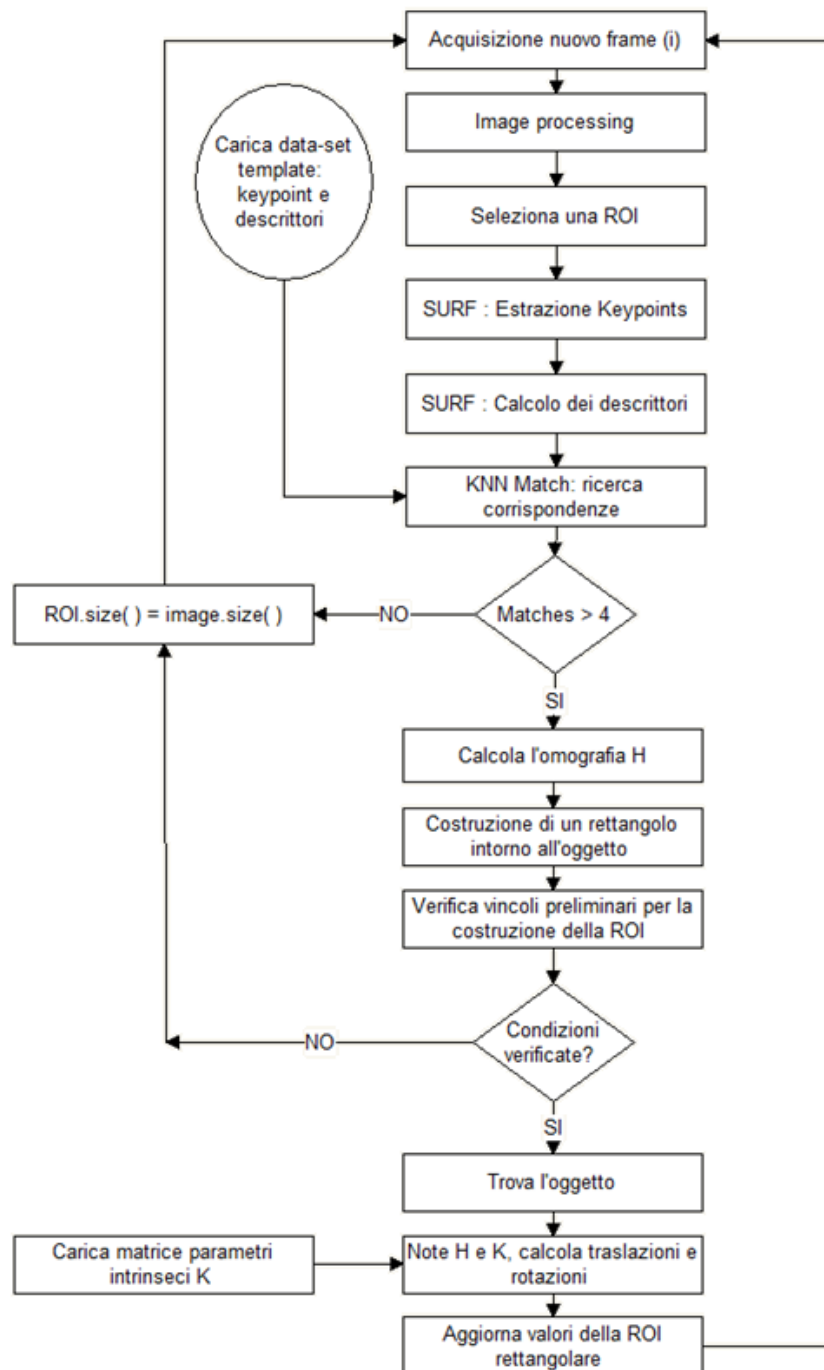
Figura 4.5 – Diagramma di flusso del programma di tracking.

Utilizzando l'omografia appena calcolata, si proiettano nel frame corrente i vertici del marker in memoria per disegnare un rettangolo colorato, che identifica l'oggetto riconosciuto.
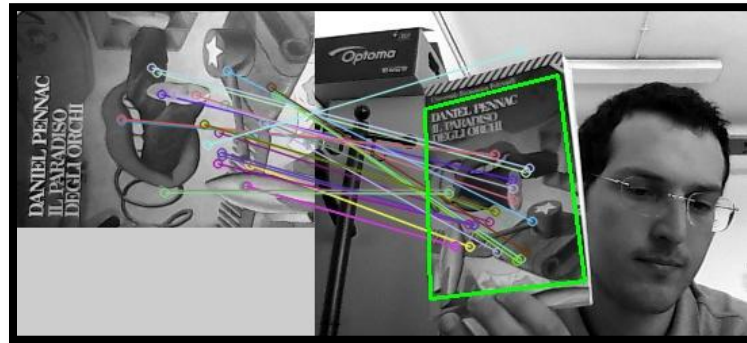
Figura 4.6 – Esempio di identificazione e matching di natural features.

Utilizzando la teoria relativa alla calibrazione della telecamera, esposta nel capitolo 1, è possibile calcolare la matrice dei parametri estrinseci, o di rototraslazione, che rappresenta la posizione della telecamera nello spazio o, viceversa, la posizione e l'orientazione dell'oggetto all'interno della scena nei 6 gradi di libertà del sistema di riferimento camera, a partire dalle matrici di omografia e dei parametri intrinseci[56]. Per 6 g.d.l. si intendono, come precisato nel capitolo 1, tre traslazioni e tre rotazioni lungo i tre assi principali di riferimento, come mostrato in **Fig. 4.7**.
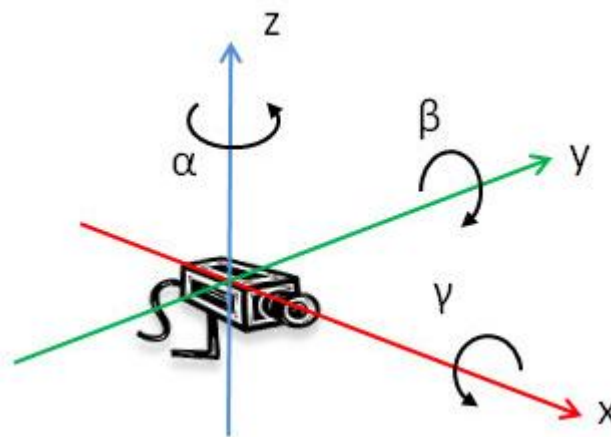
Figura 4.7 – Sistema di riferimento telecamera.

La traccia del vettore che si ottiene moltiplicando la matrice inversa dei parametri Intrinseci $[K]^{-1}$ per la prima colonna della matrice di omografia h_1 può essere utilizzata per normalizzare il prodotto tra la matrice inversa dei parametri Intrinseci e l'intera matrice di omografia $[H]$:

$$\frac{1}{\|[K]^{-1} \cdot h_1\|} \cdot [K]^{-1} \cdot [H] = [r_1 \;\; r_2 \;\; t] \tag{4.1}$$

La matrice prodotto così ottenuta contiene le prime due colonne della matrice di rotazione ed il vettore di traslazione t, che indica le coordinate della telecamera rispetto all'origine del marker[69]. La matrice di rotazione completa $[R]$ si ottiene applicando il vincolo di orto normalità fra le prime due colonne, infatti:

$$r_3 = r_1 \times r_2 \tag{4.2}$$

Affinché il vincolo imposto sia valido e per la matrice $[R]$ valga la proprietà di ortogonalità, per cui:

$$[R]^T[R] = [I] \tag{4.3}$$

è necessario scomporre $[R]$ in due matrici ortonormali $[U]$ e $[V]$, seguendo il metodo SVD[12], e ricomporre nuovamente le due matrici nella matrice di rototraslazione finale:

$$[R] = [U][I][V] \tag{4.4}$$

Componendo, infine, $[R]$ e t in un'unica matrice ordinata per colonne, si ottiene la matrice dei parametri Estrinseci, cioè di rototraslazione del marker nel sistema di riferimento della telecamera.
Si è scelto, infine, di utilizzare una *Region of Interest* (ROI) all'interno dell'immagine, nella quale ricercare il marker riconosciuto nel frame successivo. In questo modo è possibile risparmiando tempo computazionale, utilizzando le coordinate nel piano immagine del rettangolo che racchiude il marker naturale incrementate di un certo valore percentuale, per assicurarsi che l'oggetto si trovi sempre all'interno della ROI, nonostante il movimento della telecamera fra due frame successivi. È stato necessario imporre alcuni vincoli da applicare alla ROI per poter circoscrivere sempre l'oggetto, anche durante un repentino movimento della telecamera, senza correre il rischio che l'oggetto uscisse dal campo visivo. Quando l'oggetto viene identificato ed è completamente all'interno dell'immagine, la ROI corrisponde al rettangolo che individua l'oggetto, incrementato del 40% in altezza e larghezza; se parte dell'oggetto si trova al di fuori dell'immagine, la ROI che viene presa in considerazione è quella parte dell'immagine in cui compare l'oggetto.
Se, invece, il marker non viene riconosciuto o se l'acquisizione viene scartata, la ROI viene a coincidere con l'intera immagine, poiché si presenta la necessità di un'analisi più accurata nel frame successivo.
Il calcolo della ***camera pose*** avviene, infine, caricando i parametri intrinseci, output del processo di calibrazione della telecamera, e applicando le equazioni (4.1-4), descritte in precedenza. Le coordinate individuate verranno comunicate al modulo Decision Support System, presentato più avanti, per poter essere utilizzate nell'esecuzione degli altri processi del sistema.

4.2.2. Funzioni di Image Processing

Le funzioni descritte in **Tabella 4.1** sono state sviluppate pensando alla necessità di programmazione del sistema attraverso l'ambiente di editing nella *fase istruttore*. Ad ogni operazione primitiva, infatti, sono associati diversi output, a seconda degli input inseriti dall'utente. Si è scelto di sviluppare 6 funzioni principali che consentono di gestire un'ampia casistica operativa senza grandi limitazioni.
Le funzioni sono state implementate utilizzando il linguaggio C/C++ e le librerie OpenCV e DOT.
Ogni operazione viene presentata come una black box con input ed output, come illustrato nelle immagini nel seguito. Gli input, selezionati dall'istruttore del sistema, permettono il controllo dell'operazione. Gli output di ogni singola funzione saranno interpretati, invece, dal DSS per operare il controllo interattivo che il sistema si propone. Nel seguito del paragrafo saranno descritte una per una le 6 operazioni logiche che sono state scelte e saranno mostrate alcune immagini per meglio chiarirne il funzionamento pratico durante l'esecuzione dell'applicazione.

`PosE` è l'operazione che, noto il marker da individuare nella scena, identifica la posizione dell'operatore nello spazio. A livello software, questa funzione implementa lo stesso algoritmo impiegato nel tracking (**Fig. 4.5**). La differenza principale sta nella conclusione dell'operazione. Ad ogni iterazione del programma viene salvata in un vettore la matrice di rototraslazione della telecamera calcolata dall'algoritmo. I dati acquisiti vengono poi filtrati utilizzando una media mobile sugli ultimi tre valori delle traslazioni e delle rotazioni lungo i tre assi principali e le coordinate correnti vengono confrontate con quelle calcolate nell'iterazione precedente.
Quando la differenza tra le coordinate eseguita fra due acquisizioni successive si stabilizza al di sotto di un valore di soglia predefinito ("variazione %", in **Fig. 4.8**), l'operazione può ritenersi conclusa.
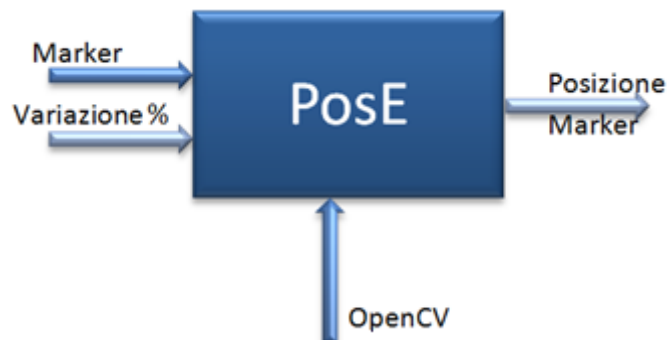
Figura 4.8 – Schema Idef A0 dell'operazione `PosE`.

Questa operazione può essere utilizzata per individuare un oggetto planare texturizzato, con lo scopo di identificare un sistema di riferimento locale per il rendering di modelli 3D nella scena.

In **Fig. 4.9** si può osservare il riconoscimento di un marker naturale, attraverso l'analisi delle corrispondenze con le features naturali. La posizione della telecamera nei 6 g.d.l è calcolata in tempo reale e viene utilizzata come input al modulo di rendering (paragrafo 4.3) per visualizzare sul display un modello virtuale della scatola.
In **Fig. 4.10** si apprezza la robustezza dell'algoritmo di tracking anche per rotazioni lungo i tre assi principali.
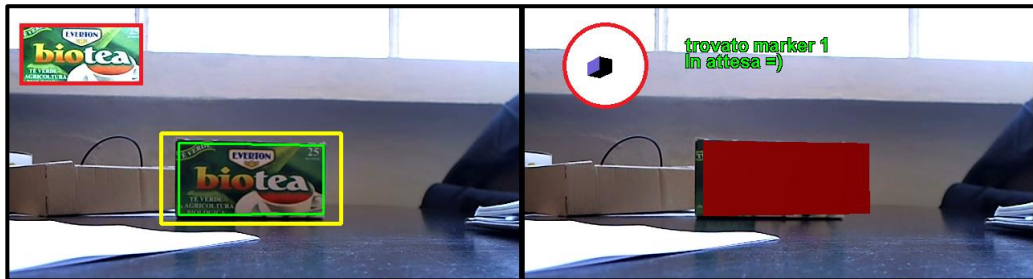
Figura 4.9 – Esecuzione dell'operazione `PosE`.
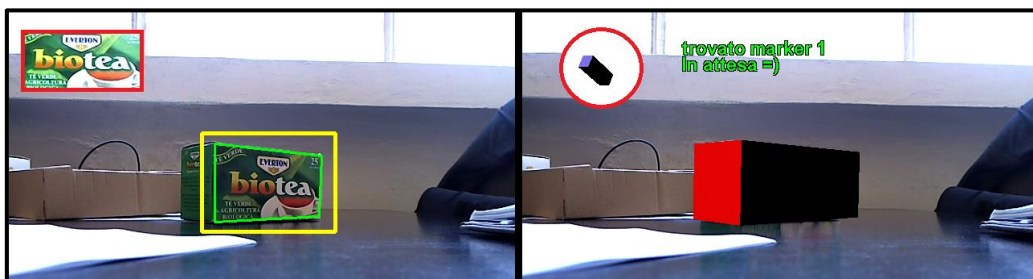
Figura 4.10 – Esecuzione del tracking anche in presenza di rotazioni.

Un altro utilizzo dell'operazione `PosE` può essere la ricerca di particolari features naturali per valutare la presenza o la posizione nella scena di un particolare oggetto con cui l'utente deve interagire. Ad esempio, questa operazione potrebbe essere impiegata per individuare un pannello da rimuovere o un'etichetta da applicare.

`TMatch` è l'operazione che si utilizza per riconoscere un oggetto di cui si è registrato il template nella fase di editing off-line. Rileva se l'oggetto sia stato effettivamente riconosciuto all'interno della scena (true/false) e fornisce le coordinate del piano immagine in cui è stato identificato l'oggetto. Nel caso in cui la procedura correntemente in uso utilizzi i template di più oggetti, l'operazione darà come output anche l'ID univoco del template riconosciuto.
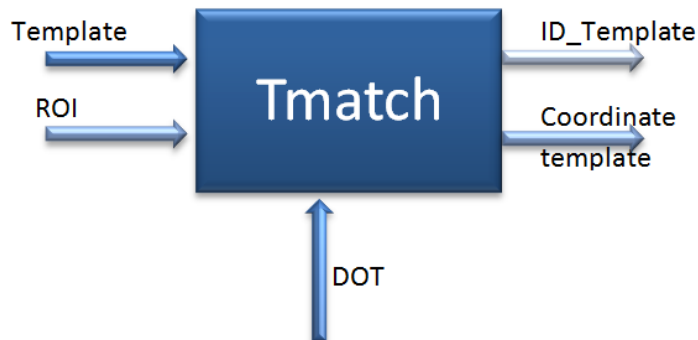
Figura 4.11 – Schema Idef A0 dell'operazione `Tmatch`.

In **Fig.4.12** è possibile osservare il funzionamento dell'algoritmo di pattern matching utilizzato. Nella *fase istruttore* si registra un template di ciascun oggetto, mostrato nel riquadro rosso in alto a sinistra dell'immagine. Durante l'esecuzione nella *fase operatore*, la funzione implementata identifica ciascun oggetto e restituisce il suo ID (come indicato dal testo stampato a video) e la sua posizione nell'immagine.
L'operazione `TMatch` presenta varie possibilità d'impiego e di funzionalità grazie alla flessibilità dei parametri che possono essere configurati durante la *fase istruttore*.

Figura 4.12 – Esecuzione dell'operazione `Tmatch`.

Una prima funzionalità è l'identificazione di un particolare oggetto all'interno della scena inquadrata, per indicarne all'utente la rimozione (attraverso il rilevamento della transizione presente/non presente) o il posizionamento in una particolare regione dello spazio.

L'operazione `TMatch`, però, si presta particolarmente bene nelle operazioni di diagnostica e di monitoraggio degli errori:

- Registrando i template dello stesso oggetto in diverse posizioni essa può fornire all'utente informazioni sul corretto posizionamento dello stesso o, se si trattasse di un componente meccanico, sulle modalità di afferraggio;
- Utilizzando template di oggetti diversi, questa operazione permette al sistema di individuare oggetti in modo selettivo, informando l'utente con quale oggetto sta compiendo l'azione indicata e se ha selezionato l'oggetto corretto;
- Miglioramento della qualità dell'interazione tra il sistema e l'utente, permettendo la comunicazione con l'operatore. Ad esempio, se l'utente deve scegliere tra diversi componenti meccanici o tra diversi utensili, può mostrare al sistema di visione gli oggetti per avere un riscontro sulle caratteristiche dei pezzi classificati;
- Selezione di un oggetto tra un insieme di potenziali scelte. Impostando un'area predefinita del campo visivo del sistema di visione come area di selezione, l'utente può spostarvi all'interno uno specifico oggetto per confermarne al sistema la scelta.

`Follow` è l'operazione che si può utilizzare, invece, per seguire il movimento di un oggetto all'interno di un volume di spazio. Questa operazione utilizza un template matching in tempo reale. Definita la regione all'interno della quale si trova l'oggetto da inseguire e calcolate le sue coordinate 2D nell'immagine con il tracciamento del marker naturale, il sistema acquisisce un template, come nell'operazione precedente, ed informa l'utente sul movimento dell'oggetto. Quando l'oggetto dell'operazione raggiunge una posizione predefinita, l'operazione si conclude con successo.
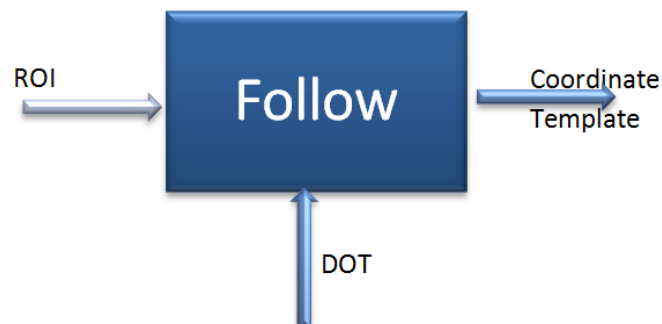
Figura 4.13 – Schema Idef A0 dell'operazione `Follow`.

La **Fig. 4.14** mostra l'utilizzo dell'operazione in combinazione con il tracking della posizione dell'osservatore, per valutare lo scorrimento del template nell'operazione di aprire il case di un personal computer. Nell'immagine a sinistra Il marker viene riconosciuto e descritto da un rettangolo verde. A partire dalla posizione del marker, viene identificata una regione di dimensioni predefinite che permette di acquisire un template da inseguire durante l'esecuzione stessa della procedura. Nell'immagine di

destra il template acquisito (contraddistinto dal colore verde) viene inseguito durante il suo movimento.

Figura 4.14 – Esecuzione dell'operazione Follow applicata allo sportello di un PC.

`BlobPG` è l'operazione che si utilizza nel caso in cui si voglia verificare la presenza/assenza o il movimento di un oggetto, di forma non definita, all'interno di una predefinita regione spaziale. Questa operazione (**Fig. 4.15**) considera la differenza fra le immagini del flusso video acquisito in real-time con un'immagine dell'area di lavoro acquisita all'avvio dell'operazione. È possibile specificare ulteriori parametri per l'identificazione di un particolare oggetto nella scena. Ad esempio, si possono definire un rapporto di forma o la dimensione massima che il particolare potrà avere, in modo da escludere falsi positivi.
Avvalendosi di alcune funzioni di blob detection, viene definita una regione all'interno della quale dovrà trovarsi il blob richiesto.
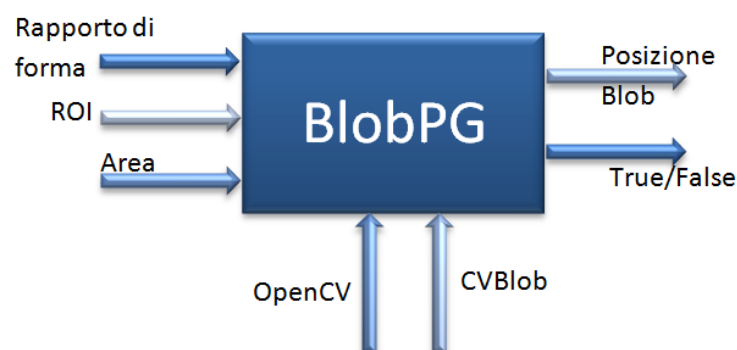
Figura 4.15 – Schema Idef A0 dell'operazione `BlobPG`.

`BlobED` è l'operazione che riconosce e misura un blob all'interno di una regione predefinita e valuta l'incremento o la diminuzione delle sue dimensioni durante lo svolgimento dell'operazione. In modo analogo alla funzione precedente, questa funzione (**Fig. 4.16**) calcola l'immagine differenza tra il flusso video corrente ed un'immagine di riferimento acquisita all'inizio dell'operazione assegnata. Identifica un blob particolare, con rapporti di forma predefiniti, e lo insegue per misurarne le variazioni di forma fino al raggiungimento dell'obiettivo.
Questa funzione può essere utilizzata per valutare l'inserimento/estrazione di un pezzo dalla propria sede o l'avvicinamento relativo di due pezzi.
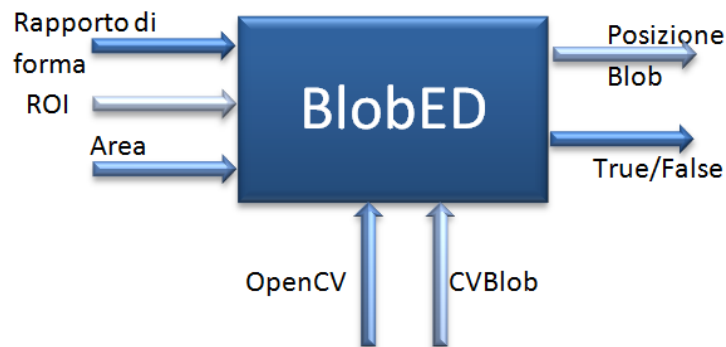
Figura 4.17 – Schema Idef A0 dell'operazione `BlobED`.

In **Fig 4.17** è rappresentata la sequenza di operazioni che entrambe le funzioni di blob detection compiono durante l'esecuzione della *fase operatore*: partendo da un frame iniziale (**a**) che mostra lo stato dell'ambiente prima dell'intervento dell'operatore, il sistema osserva le modifiche apportate all'ambiente durante l'esecuzione (**b**). Si
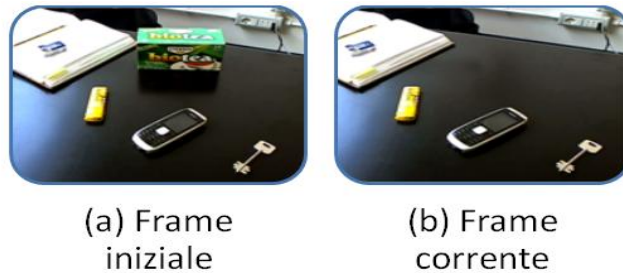
Figura 4.16 – Esecuzione dell' algoritmo di blob detection implementato nelle operazioni BlobPG e BlobED.

opera, poi, una differenza (**c**) fra il flusso video corrente ed un frame di riferimento.
All'immagine differenza risultante viene applicato un filtro binario (**d**) per estrarre regioni uniformi dell'immagine, i blob (**e**), che il sistema osserverà per compiere valutazioni in base ai parametri selezionati nella *fase istruttore*.

RGBlob è un'operazione che si può utilizzare per identificare la presenza nella scena, o in una determinata area dell'immagine, di oggetti di colore particolare e di geometria non facilmente individuabile. A differenza delle altre funzioni che sfruttano la blob detection, questa funzione si avvale dello spazio colore HSV[2] per identificare oggetti colorati. Sono stati calcolati i valori di soglia per ogni colore primario ed è stata sviluppata, a livello software, una funzione che associ tali valori alla stringa di testo relativa al colore (ad es. "verde") inserita nell'editor dall'utente.
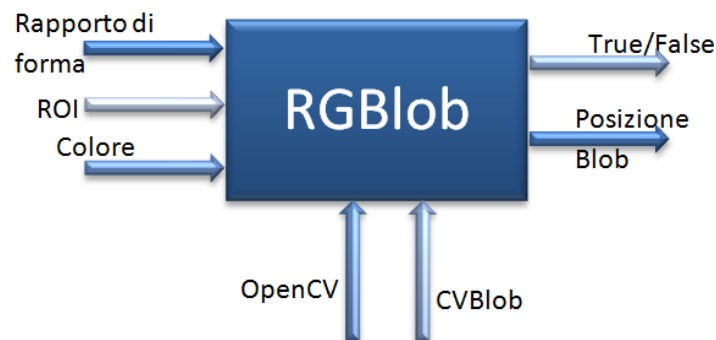
Figura 4.18 – Schema dell'operazione RGBlob.

Le potenzialità di questa funzione, oltre al rilevamento di un singolo oggetto colorato nell'immagine (ad esempio un led colorato acceso), si prestano all'individuazione selettiva di oggetti che si differenziano per il proprio colore.
Impostando, ad esempio, un'area predefinita del campo visivo del sistema di visione come area di selezione in cui l'utente può spostarvi uno specifico oggetto, il sistema può fornire un riscontro sulle caratteristiche dei pezzi classificati o indicare all'utente se il pezzo selezionato è quello corretto per l'operazione in corso.
Esempi di applicazioni possono essere la selezione di un oggetto tra un insieme di potenziali scelte o il riconoscimento di un particolare colore nella scena.

---

[2] **HSV** è l'acronimo di ***H**ue **S**aturation **V**alue* e indica sia un metodo additivo di composizione dei colori che un modo per rappresentarli in un sistema digitale. Viene anche chiamato **HSB,** ***H**ue **S**aturation **B**rightness* (tonalità, saturazione e luminosità). Il modello HSB è particolarmente orientato alla prospettiva umana, essendo basato sulla percezione che si ha di un colore in termini di tinta, sfumatura e tono. Il sistema di coordinate è cilindrico e il modello HSB è definito come un cono distorto all'interno del cilindro. La tonalità **H** viene misurata da un angolo intorno all'asse verticale, con il rosso a 0 gradi, il verde a 120 e il blu a 240. L'altezza del cono rappresenta la luminosità (**B**) con lo zero che rappresenta il nero e l'uno il bianco. La saturazione (**S**) invece va da zero, sull'asse del cono, a uno sulla superficie del cono.

4.3. Augmented Reality Module

Per renderizzare un modello virtuale all'interno del flusso video si devono conoscere i parametri intrinseci della telecamera (output del processo di calibrazione), i parametri estrinseci, ovvero la posizione del marker naturale rilevato dal tracciamento, il modello 3D da caricare ed il frame video attuale ripreso in modo da ottenere un frame video aumentato, cioè un'immagine che contenga sia la scena ripresa che l'oggetto renderizzato nella posizione voluta.

All'interno del flusso video acquisito dalla telecamera si ricerca il marker, la cui immagine è in memoria, per estrarre informazioni sulla posizione dell'osservatore. Il sistema di riferimento assoluto identificato viene utilizzato per ruotare e scalare in modo opportuno i modelli 3D nello spazio. Infine, i modelli 3D sono collocati nel flusso video, che viene così "aumentato" in tempo reale.

In **Fig. 4.19** si può osservare la pipeline del processo appena descritto: le operazione relative al tracciamento del marker naturale (colorate in blu) sono gestite dal modulo di Computer Vision, descritto nel paragrafo 4.2.1, mentre le operazioni proprie del rendering (colorate in rosso) sono state implementate nel modulo di Realtà Aumentata, oggetto del corrente paragrafo.
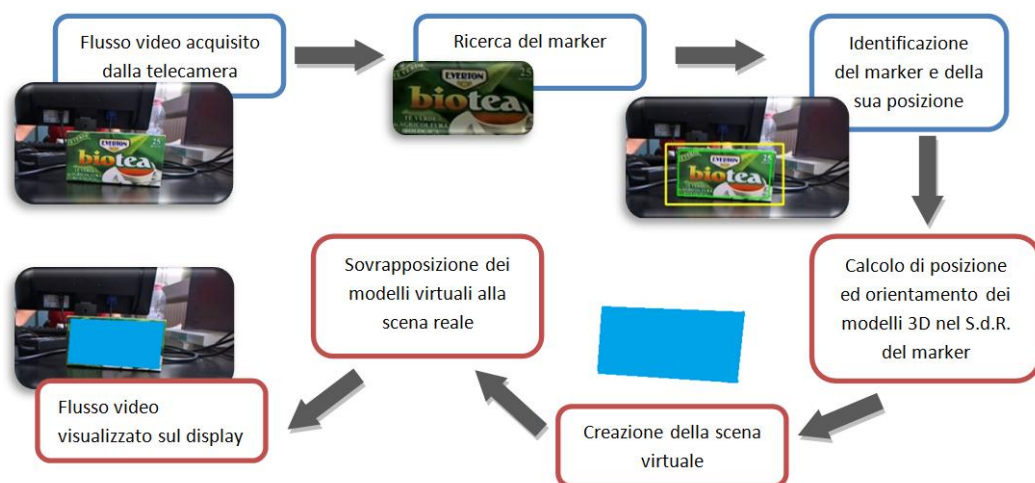
Figura 4.19 – Pipeline del processo di Rendering.

Il modulo per la realtà aumentata (ARM) ha il compito di gestire la comunicazione tra sistema ed utente attraverso la visualizzazione delle istruzioni su di un display. Come si osserva in **Fig. 4.20**, il modulo ARM è composto da tre elementi, ciascuno dei quali controlla una specifica operazione.

Il motore di Rendering (o **Rendering Engine**) gestisce la creazione della scena virtuale: riceve istruzioni dal controllo su quali modelli caricare e come visualizzarli a video. Il modulo di registrazione della scena (o **Object Registration**) interpreta i dati sul tracciamento real-time dal modulo di visione e aggiorna la posizione dei modelli all'interno della scena in tempo reale, per una corretta sovrapposizione tra il reale ed il virtuale.

Il modulo relativo all'interfaccia utente, o **GUI**, invece, si occupa di gestire l'intero flusso di informazioni grafiche da fornire all'utente.

Figura 4.20 – Modulo di realtà aumentata.

4.3.1. Rendering Engine

Per renderizzare un modello 3D caricato in memoria è stata creata una classe al cui interno vengono richiamate alcune delle funzioni presenti all'interno delle librerie GIOVE, OpenGL e OpenCV (Appendice B).
Il processo di rendering sviluppato si avvale della tecnica del *render to texture*, che permette di renderizzare una scena 3D ed utilizzarla come texture successivamente.
Per eseguire il rendering di un qualsiasi oggetto 3D i passi da seguire, implementati nel software sviluppato, sono i seguenti (**Fig. 4.22**):
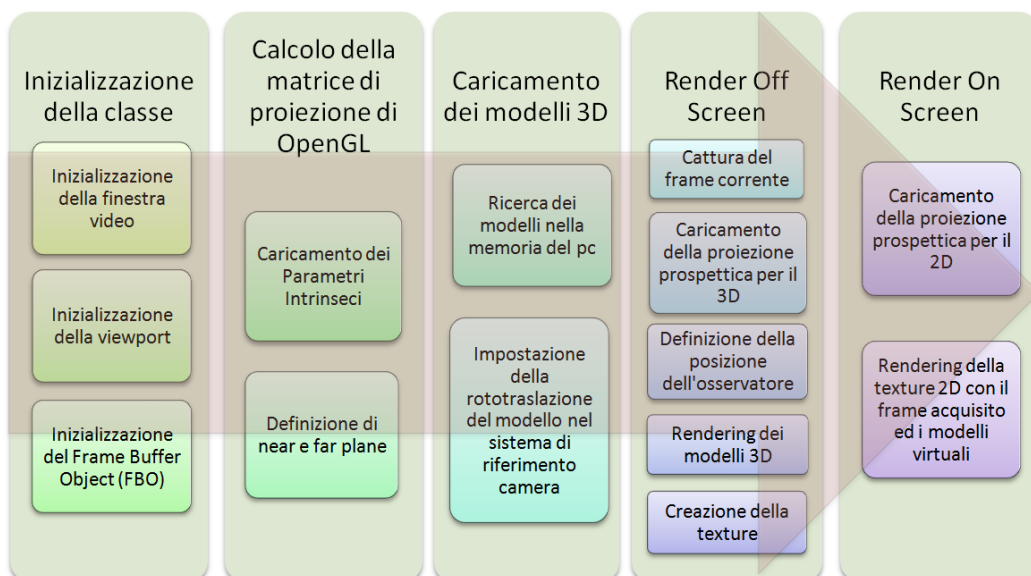
Figura 4.21 – Flusso logico del motore di rendering.

1. Inizializzazione della classe: gli argomenti del costruttore sono la larghezza e l'altezza del frame immagine acquisito dalla videocamera. Questi valori verranno utilizzati per inizializzare il *frame buffer object*[3], la *viewport* e per definire la grandezza dell'area di memoria utilizzata per contenere l'immagine renderizzata. Essendo un'applicazione real-time, il buffer di memoria destinato a contenere l'immagine renderizzata è fisso e viene dichiarato una volta sola.
2. Calcolo la matrice di proiezione di OpenGL: utilizzando il metodo `LoadProjectionMatrix` è possibile calcolare la *projection matrix* dai parametri intrinseci e dal *near* e far *plane*, i limiti di profondità dello spazio 3D renderizzato. La matrice di proiezione risultante è definita in **tabella 4.2**[59].

$\frac{2 \cdot f_x}{width}$	0	$2 \cdot \left(\frac{c_x}{width}\right) - 1$	0
0	$-\frac{2 \cdot f_y}{height}$	$2 \cdot \left(\frac{height - c_y}{height}\right) - 1$	0
0	0	$-\frac{far + near}{far - near}$	$-2 \cdot \frac{far \cdot near}{far - near}$
0	0	-1	0

Tabella 4.2 – OpenGL Projection Matrix.

3. Caricamento dei modelli 3D: tramite il metodo `LoadModel` viene aggiunto alla classe `gvGroup` (la scena che contiene tutti i modelli degli oggetti) un nuovo `gvGroup` contenente il modello 3D da renderizzare. Per renderizzare ogni oggetto bisognerà fornire i parametri estrinseci, ovvero la rototraslazione dell'oggetto rispetto al sistema di riferimento camera, ed un fattore di scala (se si vogliono ridurre o aumentare le dimensioni dell'oggetto). Inoltre, per facilitare la ricerca in memoria dei modelli caricati è possibile assegnare ad ogni gruppo un nome.
4. Render Off-Screen: utilizzando il metodo `GrabImg` si cattura l'immagine acquisita dalla webcam e la si salva nel Frame Buffer Object. A partire dai dati in uscita dal modulo del tracciamento, si definisce la posizione dell'osservatore.
   Dopo che tutti i modelli sono stati caricati e si è scelto quali rendere visibili, si genera la texture 2D contenete l'immagine acquisita ed i modelli virtuali.

[3]Il frame buffer object (FBO) è una funzionalità di OpenGL per realizzare il *render to texture*. Normalmente il rendering viene fatto su un framebuffer predefinito, detto anche *window-system-provided* framebuffer, che viene creato e gestito interamente dal sistema ed il cui contenuto è visualizzato direttamente a video. Utilizzando l'FBO si indirizza il rendering su un *application-created* framebuffer che è controllato direttamente da OpenGL ed il cui contenuto non viene direttamente visualizzato a video ma viene usato per successive elaborazioni

L'intero processo del Render Off-Screen non viene mai visualizzato a video, si lavora, infatti, su un buffer di memoria diverso dal framebuffer standard.

5. Render On-Screen: l'ultimo passo del processo si occupa della visualizzazione delle immagini.
   Si renderizza a video la primitiva OpenGL rettangolo grande quanto l'intera immagine acquisita e su di essa si applica la texture generata al passo precedente.

Oltre ai metodi appena citati, all'interno della classe del rendering sono presenti altre funzioni per la gestione dei modelli virtuali:

`Color_Child`	Attribuisce una texture in scala RGB ad uno specifico modello o gruppo.
`Set_Modelview`	Imposta la posizione di un modello nella scena, secondo un determinato sistema di riferimento.
`VisibleHideModel`	Rende visibile o invisibile un modello, prima di renderizzare l'intera scena.
`Move_Child`	Trasla nello spazio un modello, applicando uno specifico spostamento (x,y,z).
`Rotate_Child`	Ruota nello spazio un modello, applicando una rotazione euleriana (x, y, z).
`RemoveChild`	Rimuove dalla scena uno specifico oggetto 3D.

Tabella 4.3 – Funzioni aggiuntive per il rendering di modelli 3D.

Per stampare a video qualsiasi messaggio di informazione viene utilizzata la funzione `cvPutText()` di OpenCV. Il messaggio verrà visualizzato sul frame corrente acquisito dalla webcam.

4.3.2. Interfaccia utente: GUI operatore

L'interfaccia grafica utente, o Graphical User Interface (GUI), è uno dei canali con cui si compie l'interazione fra utente e sistema[60]. Nello sviluppo del software si è tenuto conto della necessità di occludere il meno possibile la visuale dell'operatore impegnato nell'attività da monitorare[1]. L'interfaccia utente è stata sviluppata come illustrato in **Fig. 4.22**. L'intera finestra è occupata dal frame video della scena osservata. In alto a destra appariranno le informazioni testuali con la descrizione dell'operazione da compiere e le indicazioni sulla sua corretta esecuzione, con eventuali azioni correttive o informazioni aggiuntive. In alto a sinistra, nel cerchio blu, saranno, invece, visualizzati eventuali modelli virtuali di componenti o utensili che l'operatore dovrà maneggiare. All'interno della scena, in corrispondenza di aree di interesse per l'operazione corrente, sarà renderizzato un rettangolo colorato.
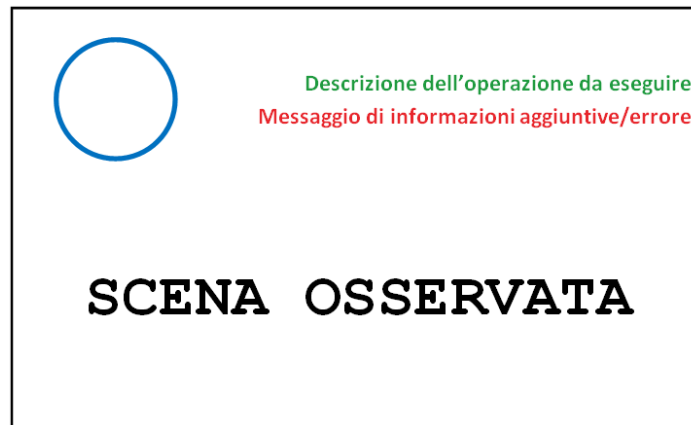
Figura 4.22 – Esempio di layout interfaccia grafica.

Per meglio focalizzare l'attenzione dell'utente si è scelto di collegare con una freccia il testo dell'istruzione dell'operazione con la regione di interesse in cui questa deve avvenire. Un esempio operativo della GUI descritta sopra si può osservare in **Fig.4.23.**

Figura 4.23 – Esempio di interfaccia utente durante l'esecuzione.

4.4. Decision Support System

Il sistema di supporto alle decisioni, o DSS, basandosi sulle informazioni raccolte dal modulo di Visione Artificiale e sulle azioni compiute dall'utente, analizza le modifiche apportate all'ambiente di lavoro e decide in quale stato il sistema debba evolvere, scegliendo fra le combinazioni possibili definite dall'utente nella fase di editing. Nel dettaglio, è possibile riassumere nei seguenti punti gli obiettivi del DSS:

- Fornire all'utente tutte le informazioni necessarie per la comprensione del compito da eseguire;
- Interpretare e riconoscere le azioni compiute dall'utente;
- Valutare lo stato di avanzamento del singolo processo ed intervenire, se necessario, per apportare misure correttive .

Affinché il DSS possa assolvere il proprio compito con efficienza è necessario definire in modo coerente e corretto i vincoli che il sistema dovrà rispettare e la sequenza di operazioni e controlli da eseguire, per osservare in modo critico l'ambiente di lavoro e ricondursi ad una casistica nota di condizioni possibili[58].
La procedura esecutiva che l'utente (fase *operatore*) carica all'avvio dell'applicazione è stata progettata nella fase *istruttore* come una sequenza di passi operativi, singole operazioni elementari che il sistema dovrà monitorare durante l'esecuzione dell'applicazione.
Ogni passo operativo è stato pensato come una struttura dati che contenga tutto il necessario per l'esecuzione della singola operazione logica ed il suo monitoraggio. Nel dettaglio, sarà costituita da:

- le descrizioni testuali relativi all'operazione che l'utente deve compiere;
- i modelli virtuali da renderizzare a video per assistere l'operatore nell'esecuzione del singolo task;
- il codice relativo all'operazione primitiva di computer vision da eseguire per monitorare l'operato dell'utente;
- gli input necessari all'esecuzione in automatico degli algoritmi di visione selezionati per l'operazione corrente;
- tutti i valori ammissibili per le variabili da monitorare, a cui saranno associate le possibili configurazioni di completamento del passo operativo all'interno della procedura.

Il funzionamento logico del DSS e le sue relazioni con gli altri moduli del sistema, che si possono osservare in **Fig. 4.24**, saranno descritti nel seguito del paragrafo.
Il DSS legge un passo operativo alla volta, ne analizza i dati presenti all'interno e li confronta con quelli restituiti dal modulo di visione artificiale, al quale fornisce indicazioni su quali operazioni svolgere e quali variabili considerare. Il flusso video in ingresso, catturato dalla telecamera, viene elaborato utilizzando le informazioni immagazzinate nel database e le funzioni di Computer Vision selezionate.

Le informazioni estratte dal frame sullo stato dell'esecuzione saranno restituite dal CVM al DSS, che le confronterà con le condizioni disponibili all'interno della struttura del passo operativo, definite nella fase di editing, per analizzare e comprendere lo stato di avanzamento dell'esecuzione e la sua correttezza.
Sulla base del riscontro di una qualsiasi condizione codificata il DSS indica al modulo di realtà aumentata (ARM) quali istruzioni e modelli 3D visualizzare a video per interagire con l'utente. Quando il DSS riconosce che la condizione di corretta esecuzione è verificata ne informa l'utente ed avanza in automatico nella procedura, caricando dalla memoria il passo operativo successivo.
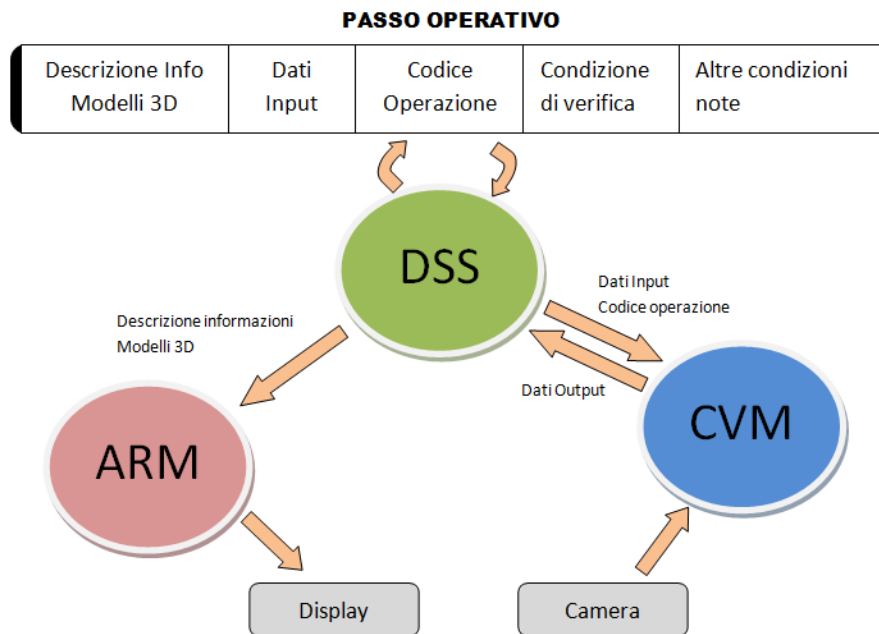
Figura 4.24 – Interazione del DSS con gli altri moduli del sistema.

Lo sviluppo dell'algoritmo del sistema di gestione dello scambio dati e di supporto alle decisioni è espresso in **Fig.4.25**. Nel diagrammare la sequenza implementata si è scelto, per chiarezza espositiva, di colorare in modo diverso le varie operazioni che il DSS compie, in relazione ai diversi moduli a cui si rivolge. Nello specifico, le operazioni di caricamento dati dal database sono individuate dal colore giallo, le operazioni che richiedono funzioni implementate nel modulo di Visione o in quello di Realtà Aumentata sono colorate, rispettivamente, in blu e rosso. Infine, le funzionalità specifiche del DSS, quali la lettura e la decodifica del passo operativo, sono messe in evidenza con il colore verde.
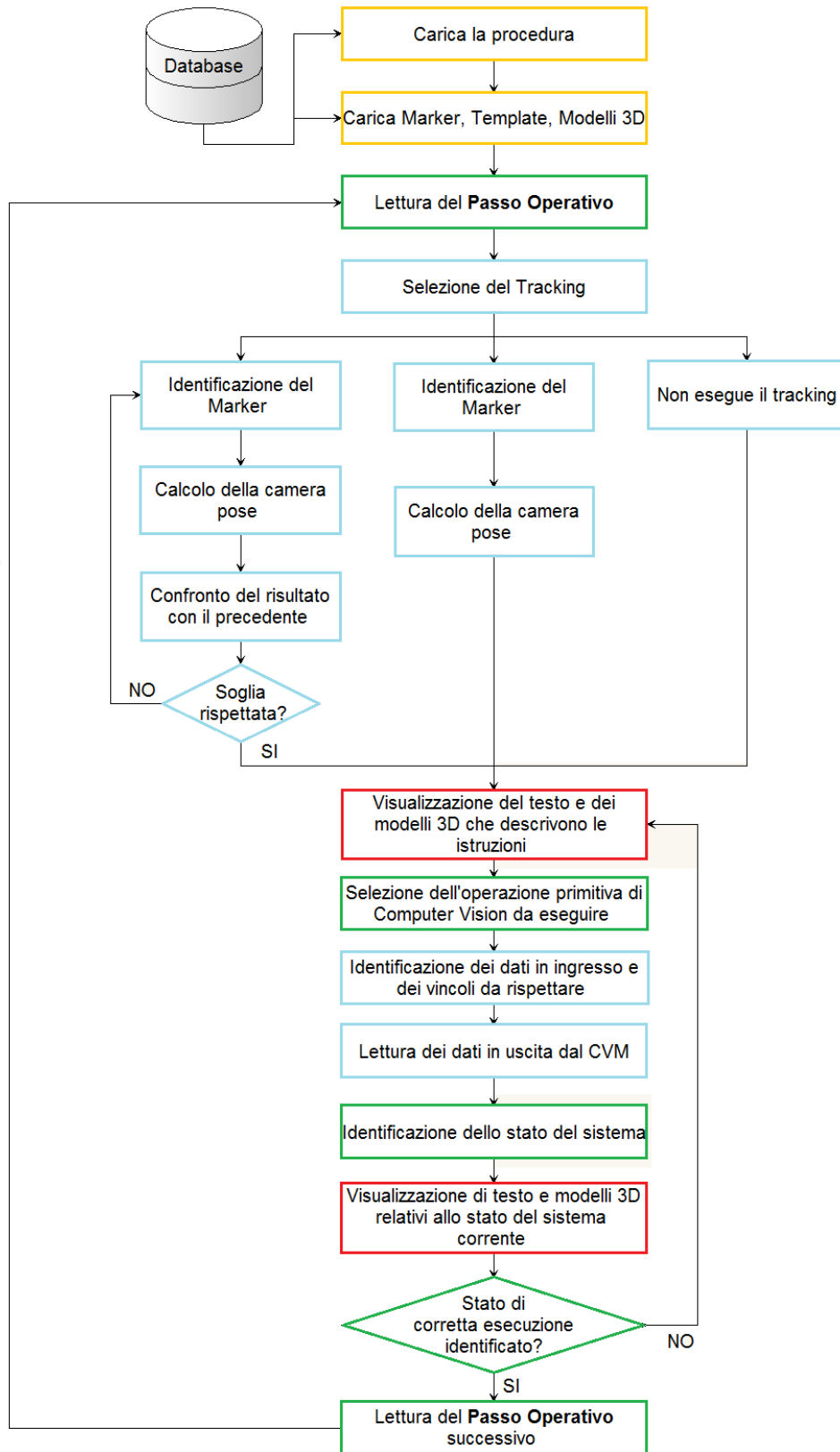
Figura 4.25 – Algoritmo di controllo operativo e supporto decisionale.

Il DSS carica il Database relativo alla procedura ed il materiale di supporto, indicato in essa, relativo alla sua esecuzione: carica dalla memoria del computer i file con i template per il pattern matching 3D, i modelli virtuali per le tecniche di Realtà Aumentata e le immagini dei marker da utilizzare come riferimento per il tracciamento. Viene poi effettuata la lettura, una riga alla volta, della tabella che contiene le regole di controllo per la procedura da monitorare.
A livello di codice, il database viene caricato utilizzando il controllo *Microsoft ActiveX Data Objects* (ADO).
La prima informazione che il DSS estrae dal passo operativo è quella relativa al tracciamento del marker naturale. È stato implementato uno `switch` per il controllo del sistema di tracciamento (**Fig. 4.26**) che permette la selezione, attraverso un parametro da passare in ingresso al CVM, di due diversi metodi per l'identificazione della posizione della telecamera. Poiché la strategia di selezione, identificazione e riagganciamento periodico ai *marker naturali* presenti nella scena dipende intrinsecamente dal tipo di applicazione in esame, si è riscontrato il bisogno di introdurre la possibilità di selezionare la tipologia di tracciamento più adatta. Vi saranno applicazioni in cui la posizione della telecamera resterà costante nel tempo, o per le quali la conoscenza in tempo reale del sistema di riferimento potrà essere superflua, ed altre in cui, invece, la telecamera sarà mobile e si riterrà necessario l'aggiornamento continuo della sua posizione nello spazio.
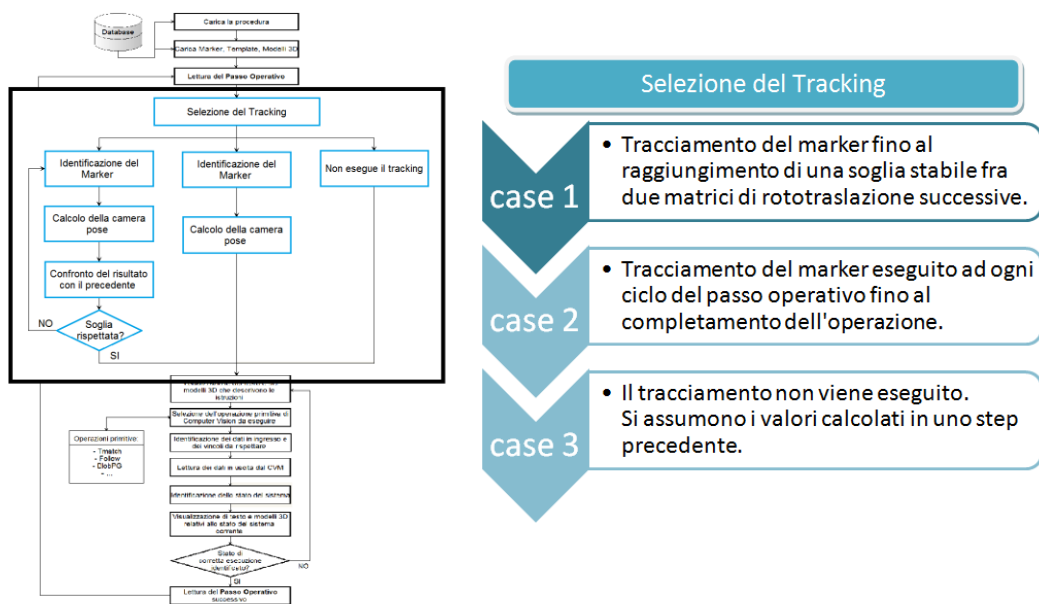
Figura 4.26 – Switch per la selezione del Tracking.

Durante la fase di istruzione del sistema sarà possibile selezionare il tipo di tracciamento richiesto per l'operazione desiderata. Le possibili opzioni (**Fig. 4.26**) sono:

1. Calcolo della *camera pose* prima di eseguire il passo operativo. Il sistema ricerca i marker nella scena e ne calcola la posizione, salvando in memoria la matrice dei parametri estrinseci risultante. L'operazione di tracking è ripetuta fino a quando la differenza fra i valori di traslazione (valori sulla quarta colonna) di due matrici successive differiscono tra loro per meno di un valore percentuale stabilito dall'utente. Il tracking non verrà eseguito fino al termine del passo operativo corrente;
2. Calcolo della *camera pose* ad ogni iterazione dell'algoritmo. Il marker viene riconosciuto in tempo reale ed i parametri estrinseci calcolati vengono passati direttamente agli algoritmi di visione e Realtà Aumentata;
3. Si utilizzano i parametri estrinseci calcolati per il passo operativo precedente.

Il modulo di Computer Vision, dopo aver eseguito il tracciamento richiesto, restituisce al DSS le informazioni sulla posizione dell'osservatore. Queste verranno inviate, insieme ai modelli 3D e alle stringhe di testo delle istruzioni, al modulo di Realtà Aumentata che si occuperà del Rendering e della visualizzazione a video per l'operatore.
Il DSS continuerà ricercando all'interno della struttura del passo operativo le informazioni sulle operazioni da monitorare.
Si è scelto di implementare, a livello software, un elenco di costanti simboliche per la selezione delle operazioni e la gestione dello scambio dati tra CVM e DSS.
Si utilizza, infatti, un secondo `switch` per richiamare l'algoritmo di visione desiderato, selezionando il codice dell'operazione secondo quanto espresso nel passo operativo in fase di editing (**Fig. 4.27**). In base ai valori contenuti nel campo *input* del database relativo alla procedura da eseguire, vengono interpretate le istruzioni da visualizzare e le condizioni da controllare ed inviate al CVM per l'esecuzione algoritmica. Per ogni modifica apportata dall'operatore all'ambiente monitorato, il CVM individua e calcola le variazioni subite dai parametri di controllo scelti e restituisce al DSS tali valori.

Il DSS confronta, poi, il valore assunto dai parametri di controllo con le possibili alternative codificate nel passo operativo alla ricerca di una condizione di verifica, che definisca lo stato di completamento dell'operazione. A seconda dello stato riconosciuto il Sistema di Supporto Decisionale ordinerà all'ARM quali modelli e testo visualizzare sul display dell'utente.
Queste operazioni vengono compiute ciclicamente dal DSS fino al raggiungimento di una condizione di verifica, con la quale si accede allo stato di completa esecuzione del singolo passo operativo. Dopo un certo tempo di attesa per la comunicazione con l'utente, il DSS procede alla lettura del passo operativo successivo, ripetendo tutto il ciclo appena descritto (**Fig. 4.25**).
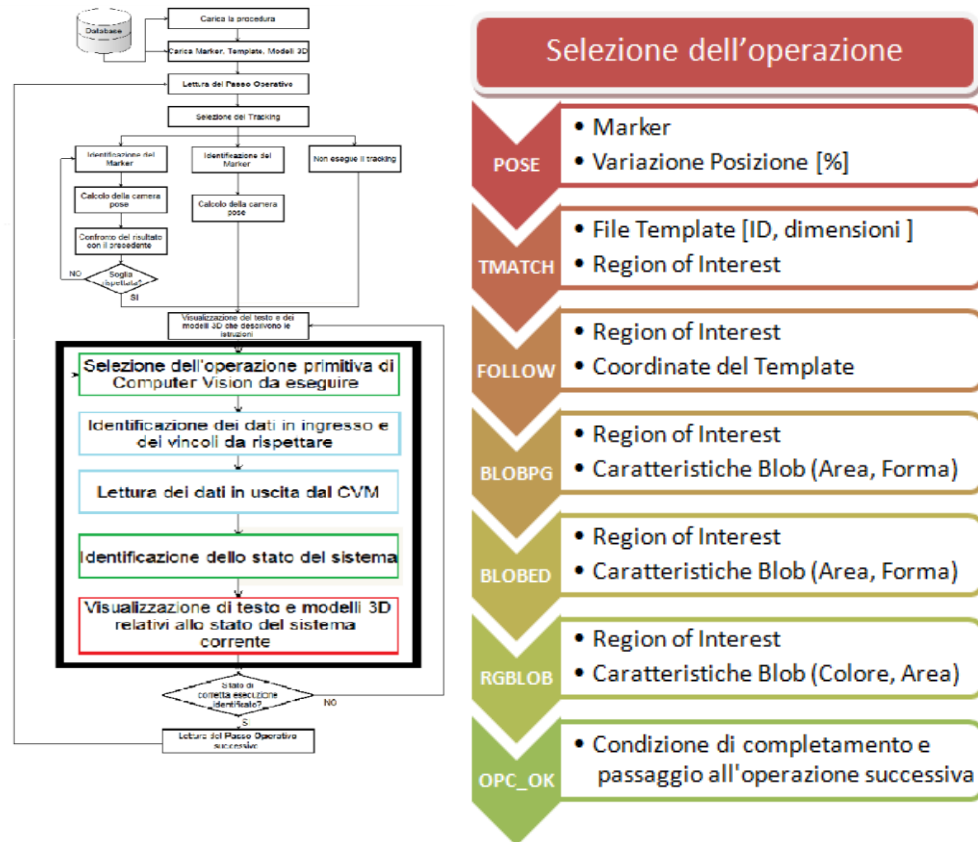
Figura 4.27 – Switch per la selezione dell'operazione e dei parametri da fornire in ingresso al CVM.

4.5. Database

Le regole che determinano il comportamento del sistema e l'utilizzo delle diverse funzionalità implementate vengono codificate all'interno di una tabella relativa alla singola procedura da eseguire.
La tabella della procedura viene creata dall'istruttore del sistema attraverso l'apposito ambiente di editing ed è salvata, insieme al materiale di supporto necessario al suo svolgimento in automatico, all'interno di un database. Tale database conterrà, inoltre, l'insieme di tutte le procedure che possono essere monitorate dal sistema ed i loro file di supporto.
L'istruttore del sistema, conoscendo a priori la procedura che l'operatore dovrà eseguire ed i potenziali errori che potrà commettere, ha il compito di redigere una lista di operazioni e controlli che il sistema deve visualizzare ed elaborare. Questo si rende necessario in quanto l'utente finale, nella fase operativa, deve essere tanto informato quanto controllato sulle operazioni che sta compiendo. Il sistema, in modo

automatico, dovrà guidare l'operatore nell'avanzamento della procedura fino al corretto completamento dei compiti dell'operatore, intervenendo laddove riscontri imperfezioni ed errori.
La singola procedura si articola in più elementi caratteristici. La procedura è una sequenza ordinata di passi operativi che, come già espresso nel paragrafo 4.4, rappresentano singolarmente un'operazione elementare da eseguire e le informazioni necessarie al suo monitoraggio.
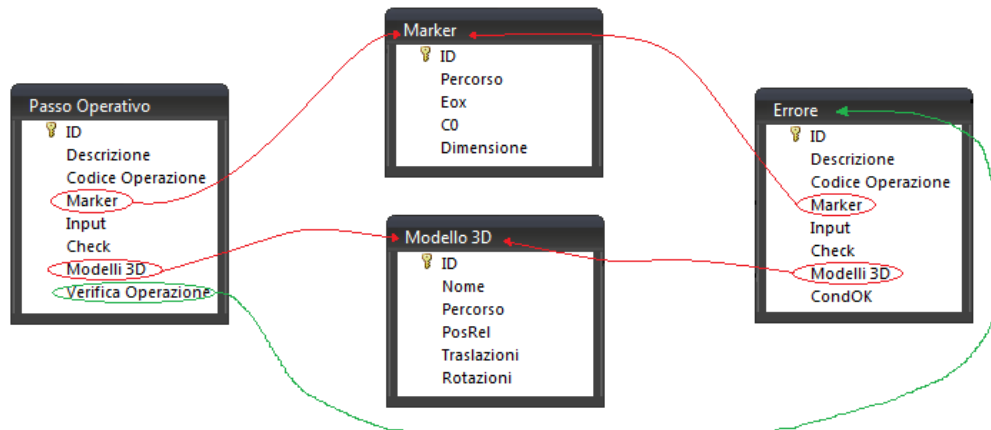
Figura 4.28 – Struttura del database.

Ad ogni passo operativo, inoltre, sono associate più strutture (tabelle) ausiliarie che raccolgono maggiori dettagli su elementi specifici del sistema, quali Marker e modelli tridimensionali. La **Fig. 4.28** mostra le relazioni fra le strutture che compongono la tabella che contiene la sequenza di passi operativi, o Procedura, a cui l'applicazione accede. Gli elementi appartenenti a ciascuna struttura saranno descritti nel seguito del paragrafo con maggiore dettaglio.

4.5.1. La Procedura

La tabella *Procedura* è l'oggetto della fase di training off-line (fase istruttore) e sarà la base per l'esecuzione on-line delle attività dell'operatore (fase operatore).
Ogni riga di questa tabella individua l'operazione per la quale il sistema di controllo dovrà verificare l'esatta esecuzione e le diverse opzioni associate ad essa quali, ad esempio, la tipologia del modello 3D da visualizzare o le condizioni per cui l'operazione può ritenersi conclusa.
Nel dettaglio, ogni riga, o passo operativo, contiene le seguenti informazioni:

- **ID**: numero progressivo del passo operativo;
- **Descrizione**: testo informativo, associato alle istruzioni da compiere, che si intende visualizzare sul display;
- **Operazione primitiva**: codice alfanumerico che permette di accedere ad un elenco di operazioni primitive di computer vision, di cui si parlerà nei paragrafi

seguenti, che identificano le diverse operazioni di controllo sull'attività in esecuzione dall'operatore;

- **Marker di riferimento**: percorso dei Marker necessari per il tracciamento in real-time dell'operatore durante l'esecuzione del passo operativo relativo. A questo elemento è associata la struttura *Marker* che raccoglie informazioni di maggior dettaglio sui diversi marker naturali impiegati dal sistema;
- **Input**: raccoglie informazioni necessarie all'esecuzione del controllo. Esempi di questa condizione, suggeriti nella sezione relativa alle operazioni, possono essere un volume di spazio in cui individuare un oggetto, la distanza alla quale due oggetti dovranno trovarsi, o il nome del *template* da caricare;
- **Check**: condizione di corretta esecuzione, con la quale il sistema può passare alla riga successiva;
- **Modello 3D**: riferimento alla struttura *Modello 3D* che permette di scegliere quale, fra i modelli presenti in memoria, dovrà essere utilizzato dal sistema nel passo operativo corrente. Anche a questo elemento è associata una particolare struttura dati di riferimento;
- **Richiesta di verifica**: per alcune operazioni potranno essere richieste verifiche aggiuntive, necessarie per la corretta esecuzione dell'operazione e per rilevare quegli errori codificati, preliminarmente individuati dall'istruttore, ai quali è possibile riferirsi secondo le istruzioni riportate nella *tabella errori.* È stata adottata una variabile booleana che specifica se per il passo operativo corrente viene richiesta una verifica aggiuntiva.

4.5.2. Marker

Affinché sia possibile calcolare la posizione dell'osservatore nello spazio e collocare oggetti 3D nella scena è necessario definire, nella fase off-line di training, gli oggetti (marker) che costituiranno il sistema di riferimento dell'intera applicazione.
Le informazioni che andranno gestite per ogni marker naturale sono:

- **Nome** dell'immagine del marker;
- **Percorso** del file immagine da ricercare all'interno della memoria del computer;
- **Matrice di trasformazione** delle coordinate dal sistema di riferimento locale a quello assoluto. Il calcolo di questa matrice è un output del processo di calibrazione del Markerfield, di cui si discuterà in seguito;
- **Coefficiente di affidabilità**, necessario al tracciamento nei 6 DOF (Degree of freedom) della telecamera;
- **Dimensione** caratteristica dell'oggetto, necessaria ad estrarre anche misure di scala dall'immagine durante la procedura.

L'istruttore del sistema deve analizzare attentamente la macchina e l'ambiente in cui l'utente opererà e ricercare i possibili marker naturali da utilizzare come riferimento per le fasi di realtà aumentata e per il tracking in generale. I marker naturali possono essere delle zone planari a bordo macchina o presenti all'interno dell'ambiente. Un

esempio potrebbero essere le zone texturizzate presenti sul corpo stesso della macchina (loghi o marchi), cartelli di sicurezza, posti sempre a bordo macchina o affissi ad una parete.

4.5.3. Modelli 3D

Le informazioni sui modelli 3D da caricare sono:

- **Nome**: nome univoco per ogni modello caricato. L'applicazione utilizzerà questo dato per manipolare il modello. Per un utilizzo efficiente dei modelli virtuali si è pensato di anteporre al nome un prefisso che identifichi la tipologia del modello. Ad esempio, un oggetto 3D che sarà collocato nella scena e sarà disponibile per l'interazione avrà un prefisso diverso da un modello che servirà solo per informare, nell'interfaccia utente, sull'operazione da compiere;
- **Path**: percorso del file del modello 3D;
- **Traslazioni**: rappresentano le traslazioni (x,y,z) in mm del modello rispetto al proprio riferimento;
- **Rotazioni**: rappresentano le rotazioni lungo (x,y,z) in gradi del modello rispetto al riferimento;
- **PosRel**: questo codice indica se le rotazioni e traslazioni sono riferite al sistema di riferimento globale (= 0) ottenuto tramite il marker o relative ad un altro modello 3D caricato (= nome o ID del modello di riferimento).

4.5.4. Errori e verifiche aggiuntive

Tra le possibili condizioni di esercizio l'istruttore dovrebbe contemplare la possibilità che l'utente commetta delle imprecisioni o degli errori più o meno grossolani. Il sistema di controllo sarà tanto più preciso ed accurato nella gestione degli errori quanto più l'istruttore sarà solerte nella previsione e definizione degli errori più comuni. Inoltre, considerando le singole operazioni elementari sviluppate, potrebbe essere necessario utilizzare più di una funzione per monitorare la corretta esecuzione di una singola operazione dell'utente.

La *tabella errori* è inclusa nella *tabella procedura* e contiene le seguenti informazioni:

- **Operazione:** operazione primitiva di computer vision con la quale operare un ulteriore controllo;
- **condizione di errore:** condizione da rilevare per definire l'errore, nel caso più errori possano essere associati alla stessa operazione della procedura;
- **descrizione:** testo descrittivo dell'errore da visualizzare per informare l'utente
- **modello 3D:** modello virtuale da renderizzare (ad esempio, il modello potrebbe essere colorato in modo diverso rispetto a quello da visualizzare nel caso di operazione corretta);
- **condizione_OK:** variabile booleana necessaria per uscire dal controllo degli errori e passare al passo operativo successivo. Contrassegna la riga

contenente la condizione accettabile per concludere l’esecuzione del passo operativo corrente..

ID	Tipo	Descrizione	Operazione	Ref Marker	Input	Check	Verifica	Condizione(
1	☐	identifica marker	PosE	./T24/marker5	1	0.01	☐	☐
2	☐	rimuovi vite 1	BlobPG	./T24/pcA2.pn	1	120 65 50	☐	☐
3	☐	rimuovi vite 2	BlobPG	-1	1	120 365 50	☐	☐
4	☑	Rimuovere lo sportello	Follow	./T24/awlab.pr	-210 170	-420 170	☐	☐
5	☐	Selezionare la scheda RAM	Tmatch	./T24/fanP1.pr	./T24/1/templ	135 -20	☑	☐
6	☑	Scheda da 512 MB	Tmatch	-1	-1	4	☐	☐
7	☑	Scheda da 1 GB	Tmatch	-1	-1	5	☐	☑
8	☐	Inserire la scheda	BlobPG	-1	3	50 50 40	☐	☐

Figura 4.29 – Esempio di Procedura: sostituzione di una scheda RAM.

Questa funzionalità del sistema può essere utilizzata anche nei casi in cui l’utente sia costretto a scegliere alternativamente tra più oggetti durante l’esecuzione di un particolare compito, guidandone la selezione con informazioni aggiuntive o avvisi di errore nel caso in cui la scelta sia obbligata da vincoli imposti a priori.

4.6. Modalità Istruttore: Editing Environment

Una delle peculiarità del sistema presentato in questo lavoro di Tesi è la possibilità di programmare il sistema per diversi ambiti e situazioni applicative, senza dover intervenire con le dovute correzioni laddove si riscontrino criticità particolari, utilizzando in modo diverso tutti gli elementi che lo compongono.

L’ambiente di creazione del database relativo alla singola procedura di manutenzione o assemblaggio è stato progettato per permettere all’utente di redigere, attraverso un’interfaccia grafica intuitiva, la sequenza di operazioni che il controllo dovrà eseguire, allegando in modo semplice, ma completo, tutti i dati di cui avrà bisogno nel suo funzionamento autonomo (file da caricare, parametri da impostare, marker da riconoscere, ecc..).
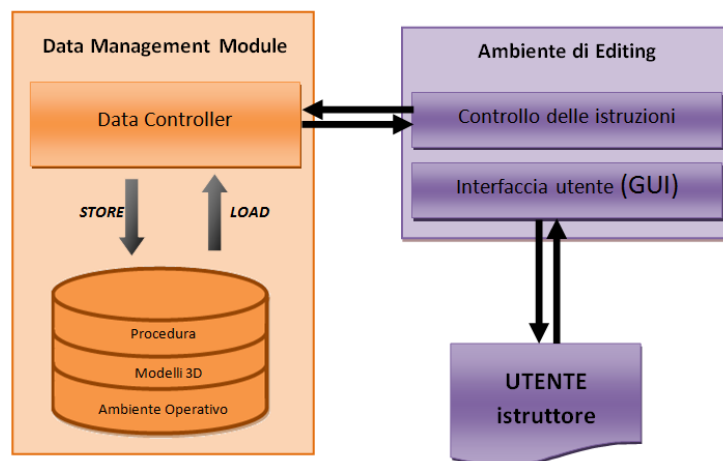
Figura 4.30 – Architettura del sistema per la fase istruttore.

Il modulo di gestione dell’editing, inoltre, controlla e segnala la presenza di eventuali errori di compilazione, suggerendone la correzione e mostrando le possibili modifiche da apportare ai parametri principali.

4.6.1. Gestione dell’interfaccia grafica

Per avviare la creazione di un nuovo database, l’utente istruttore dovrà utilizzare il file eseguibile **istruttore.exe** e seguire una procedura guidata. Nella **Fig. 4.31** è possibile osservare la schermata iniziale dell’applicazione. Nei campi di testo bisogna inserire il codice relativo alla procedura che si andrà a sviluppare ed al modello di telecamera che si sta utilizzando.
L’interfaccia utente per la creazione della procedura consente di inserire tutte le informazioni necessarie all’esecuzione all’interno di un’unica tabella e di salvare i dati ed i file da caricare successivamente in cartelle specifiche nella memoria del computer.
I primi pulsanti che si possono osservare sono quelli relativi alla gestione del database.
È possibile creare e salvare su file una nuova procedura, ma anche caricare, modificare o eliminare una procedura già esistente nel disco rigido.
Gli altri pulsanti (ultima riga in figura) serviranno ad acquisire e salvare i file necessari e consentiranno il completamento automatico della tabella della procedura.

Figura 4.31 – Finestra principale della procedura di training del sistema.

L’interfaccia è stata realizzata utilizzando il linguaggio C++ e le componenti grafiche disponibili nell’ambiente di sviluppo *Microsoft Visual Studio 2010.* Il Database è stato associato all’ambiente di editing tramite il controllo `DataGridView` di Windows Form.

La gestione del salvataggio dei diversi output dell'operazione di editing avviene attraverso un sistema di sotto-cartelle (**Fig. 4.32**) che permette di ricondursi sempre alla procedura richiesta e consente di esportare i diversi file anche su altri dispositivi di memorizzazione dati. La cartella principale avrà lo stesso nome della procedura, e tante sottocartelle, quanti saranno i passi operativi.
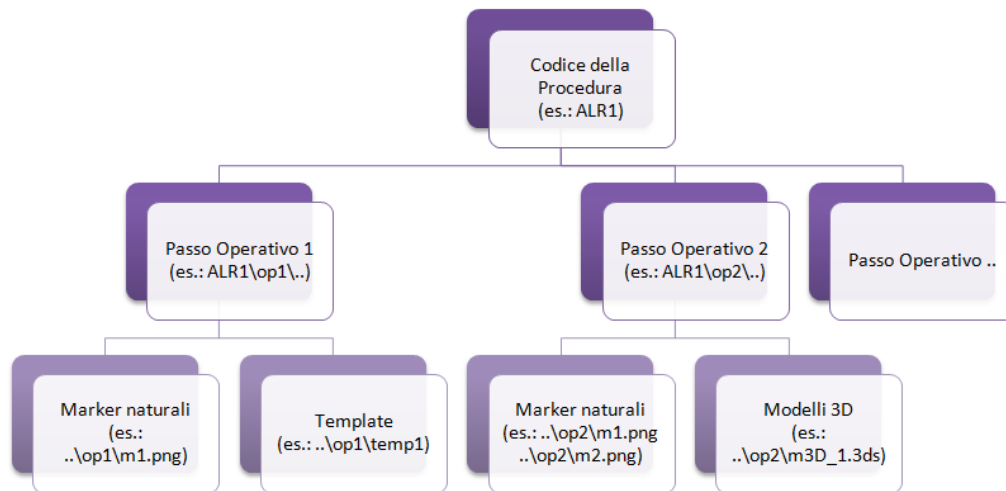
Figura 4.32 – Gestione dei dati relativi alla procedura.

In ogni cartella del passo operativo saranno contenute le immagini dei marker, i file dei template e tutto il materiale di supporto che sarà creato e gestito tramite l'interfaccia grafica della *fase istruttore* (**Fig. 4.31**).

4.6.1.1. Salvataggio dei Marker

Affinché sia possibile calcolare la posizione dell'osservatore nello spazio e collocare oggetti 3D nella scena è necessario definire, nella fase off-line di training (*fase istruttore*), gli oggetti (marker) che costituiranno il sistema di riferimento dell'intera applicazione.

Il pulsante *Salva Marker* permette di selezionare manualmente i marker dal flusso video della videocamera. Si consiglia di inquadrare il marker da vicino, il più possibile, per consentire un migliore riconoscimento dello stesso nelle operazioni successive. Si raccomanda, inoltre, di prestare attenzione alle condizioni di illuminazione, per evitare marker troppo, o troppo poco, esposti alla luce.

4.6.1.2. Salvataggio dei Template

I *template* rappresentano oggetti presenti nella scena con cui l'operatore dovrà interagire. A differenza dei *marker*, che sono per lo più fissi in posizioni predefinite, i template sono oggetti tridimensionali di cui non si conosce a priori la posizione

all'interno della scena. Saranno individuati da appositi algoritmi (nella fase operatore, paragrafo 4.2.2) per ricercarne la presenza o suggerire all'utente informazioni circa il loro posizionamento.
Il pulsante *Salva Template* permette di istruire il sistema sui template da ricercare. L'operazione di salvataggio dei template, per quanto sia abbastanza semplice nello svolgimento pratico, richiede una certa attenzione da parte dell'istruttore e uno studio preliminare sugli oggetti da utilizzare e sulle loro posizioni nella scena durante l'esecuzione delle operazioni.
All'avvio del programma si apre una finestra in cui è mostrato il flusso video acquisito in tempo reale dalla webcam.
All'interno della finestra è possibile, utilizzando il mouse, muovere un rettangolo giallo con il quale identificare il template. Il salvataggio del template dell'oggetto si effettua con una pressione del tasto destro del mouse. si può apprezzare immediatamente il risultato di una corretta acquisizione, osservando i contorni dell'oggetto diventare di colore verde.
Durante l'acquisizione è necessario che l'oggetto si trovi appoggiato ad una superficie di colore neutro (bianco o grigio). Inoltre, per acquisire più "viste" di uno stesso oggetto, è necessario ruotare e/o scalare (traslare allontanando la telecamera) l'oggetto, in base a come si prevede sarà utilizzato dal sistema nella fase di monitoraggio. Nelle figure seguenti si possono osservare entrambe le fasi operative di template acquisito correttamente (**Fig. 4.34**) e template riconosciuto parzialmente (**Fig. 4.33**). In questo secondo caso è necessario un nuovo salvataggio del template nella nuova posizione.
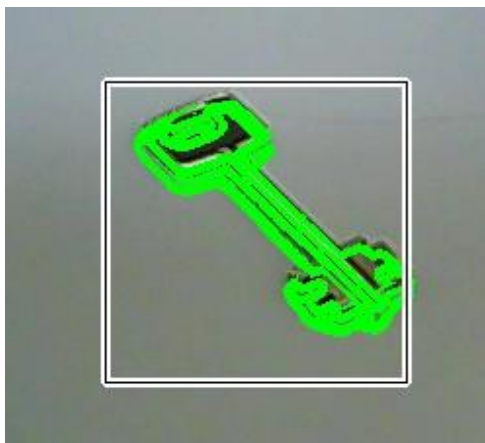
Figura 4.34 – Template acquisito correttamente.

Figura 4.33 – Template riconosciuto parzialmente.

Dopo aver salvato i template relativi ad un oggetto, è possibile chiudere il programma o acquisire i template di un nuovo oggetto. Alla fine dell'acquisizione, il programma salva i template sull'hard disk e riempie la tabella della procedura con le informazioni relative ai singoli template.

4.6.1.3. Caricamento dei Modelli 3D

Si ricorda che è compito dell'utente istruttore creare o reperire tutti i modelli 3D necessari alla procedura.
Per riempire la tabella con le informazioni appena descritte l'utente dovrà compilare i campi della finestra in basso a sinistra dell'interfaccia principale e premere il tasto relativo al caricamento di un modello. Si aprirà una finestra di ricerca, con la quale sarà possibile selezionare il modello desiderato. Alla pressione del pulsante *OK*, tutte le informazioni saranno inserite nella tabella principale e sarà possibile procedere al caricamento di un nuovo modello.

4.6.1.4. Operazioni primitive

Le operazioni di Computer Vision possono essere selezionate da un elenco predefinito, grazie ad un menu a tendina. La tabella in **Fig.4.35**, non modificabile dall'istruttore, contiene tutte le informazioni necessarie al corretto inserimento degli input relativi a ciascuna funzione nella procedura. Ad ogni riga è associata un'operazione primitiva di computer vision, una breve descrizione del suo funzionamento e una serie di vincoli e condizioni che possono essere utilizzati per verificare l'avanzamento della procedura.
L'esecuzione delle operazioni ed il loro sviluppo è demandato al modulo di *Image Processing* del CVM.

ID	Descrizione	Codice	Input	Output
1	riconosci oggetto con template noto	Tmatch	template	ROI, ID
2	seguire movimento (template on-line)	Follow	template	ROI
3	verifica presenza/assenza	BlobPG	ROI	blob
4	misura dimensione blob	BlobED	ROI	blob
5	identifica oggetto colorato	RGBlob	colore	blob, ROI
6	trova posizione telecamera	PosE	marker	pose

Figura 4.35 – Tabella delle operazioni primitive.

4.7. Calibrazione del sistema

4.7.1. Calibrazione off-line della telecamera

La calibrazione della telecamera ha luogo durante la fase di editing della procedura e viene gestita dal modulo *Editing Environment*.
Le informazioni riguardanti la telecamera utilizzata, il cui codice può essere inserito nel campo di testo in alto a sinistra dell'interfaccia, saranno salvate nel file output del processo di calibrazione. I dati relativi alla telecamera sono:

- **Risoluzione orizzontale** della telecamera in pixel;
- **Risoluzione verticale** della telecamera in pixel;
- **Parametri intrinseci** della webcam, output dell'operazione di calibrazione.

Il programma sviluppato è stato realizzato utilizzando alcune funzioni della libreria OpenCV e può essere eseguito cliccando sul bottone *Calibrazione*, presente nell'interfaccia di editing del sistema.
La procedura di calibrazione, schematizzata nel diagramma di flusso in **Fig. 4.36**, va eseguita utilizzando un pattern di calibrazione noto. Si è scelta, per i motivi esposti nel capitolo 1, una scacchiera di dimensioni 6x8 riquadri. L'immagine della scacchiera è stata stampata in alta definizione ed incollata ad un supporto rigido, per assicurarsi che tutti i punti della scacchiera giacessero sullo stesso piano durante l'acquisizione delle immagini nell'esecuzione del programma di calibrazione[70].

All'avvio, il programma di calibrazione visualizza a video due immagini. Nella finestra di sinistra è mostrato il video della telecamera, in modo che l'utente possa osservarne i movimenti e decidere da quale angolazione osservare la scacchiera. Nella finestra di destra, invece, è visualizzata l'immagine che il programma utilizza per il calcolo dei parametri intrinseci, vincolo al completamento della procedura.
Per procedere col calcolo dei parametri della telecamera è necessario avere a disposizione un numero $n \geq 3$ di immagini del pattern di calibrazione in diverse posizioni ed orientamento, in modo da coprire, con più acquisizioni successive, tutta la superficie del sensore.
La funzione `cvFindChessboardCorner` restituisce la posizione approssimata degli spigoli interni della scacchiera, detti corner, sotto forma di coordinate 2D nel piano immagine[12].
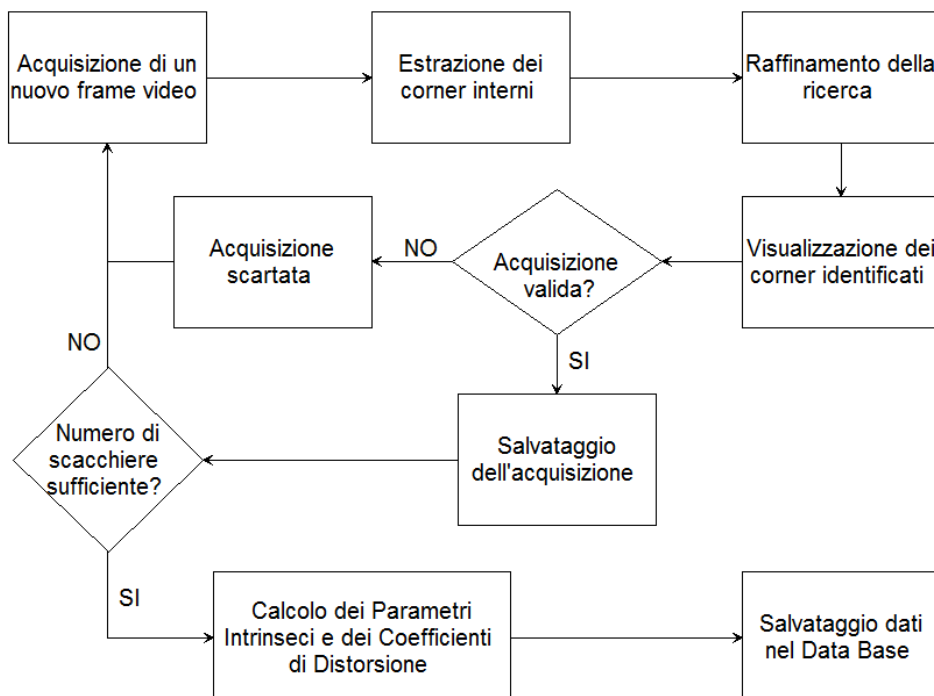
Figura 4.36 – Diagramma di flusso del programma di calibrazione sviluppato.

Per corner interni si intendono tutti i corner della scacchiera che non si trovano sul riquadro esterno: per la scacchiera 6x8 impiegata, i corner interni formano una griglia di dimensione 5x7.
La ricerca dei corner interni viene raffinata utilizzando la funzione `cvFindCornerSubPix` che, aumentando la risoluzione della ricerca, raffina l'errore di approssimazione.
Se l'immagine è stata acquisita correttamente e la scacchiera è stata riconosciuta dal programma, il programma informa l'utente sul numero residuo di scacchiere da acquisire per completare l'operazione di calibrazione e colora i corner rilevati in modo diverso per ogni riga, ad indicare l'orientamento della scacchiera rispetto al sistema di riferimento (**Fig. 4.38**). Il risultato di una corretta acquisizione, può essere visualizzato attraverso la funzione `cvDrawChessboardCorners`.
Quando l'algoritmo non riesce a rilevare il numero di spigoli interni, fornito in input alla procedura, a causa di errori o imprecisioni dell'utente, viene visualizzato un messaggio di errore, in alto a sinistra, che invita l'utente a prestare maggiore attenzione all'operazione in corso. In caso di occlusione parziale della scacchiera, a causa della presenza della mano dell'utente o dell' uscita della scacchiera dall'angolo di visione della telecamera, l'acquisizione viene ugualmente scartata e, in caso di riconoscimento parziale, i corner vengono evidenziati con il simbolo ⊗ in rosso (**Fig.4.37**).
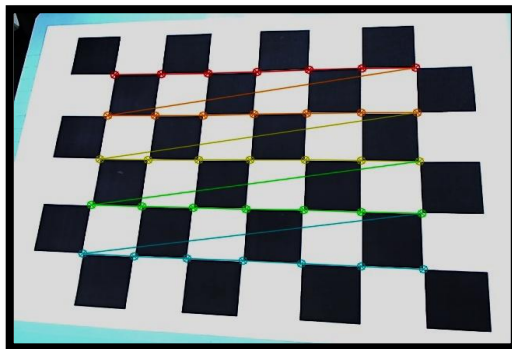
Figura 4.37 – Pattern riconosciuto correttamente.
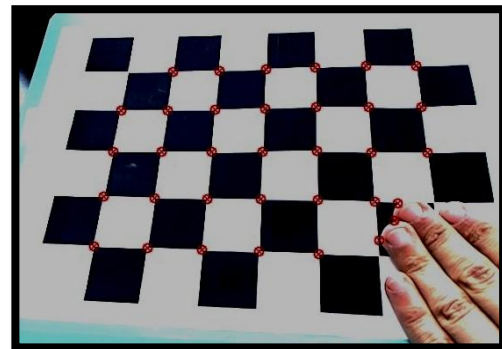
Figura 4.38 – Pattern di calibrazione scartato.

Una volta individuati e memorizzati i corner trovati in tutte le immagini si procede con il calcolo dei parametri di calibrazione descritti nel capitolo 1, utilizzando la funzione `cvCalibrateCamera2`, anch'essa presente in OpenCV. Questa funzione applica il metodo di Zhang[73] e calcola i parametri intrinseci ed i coefficienti di distorsione attraverso il legame tra i corner 2D trovati nelle immagini e i corner 3D, che vengono creati secondo la struttura della scacchiera a partire dall'origine e con le coordinate che variano in funzione della dimensione dei quadrati.

Alla fine dell'elaborazione il programma salva sul disco del computer un file contenente i parametri della calibrazione.

4.7.2. Calibrazione del sistema di riferimento globale: il Markerfield

Il sistema di tracciamento markerless sviluppato, descritto nel paragrafo ad esso dedicato (CVM), consente di calcolare in tempo reale la posizione della telecamera a partire dall'identificazione di un elemento piano e texturizzato presente nella scena e nella memoria del sistema. Il sistema, poi, collocherà i modelli virtuali basandosi su un sistema di riferimento che avrà la propria origine nel marker naturale riconosciuto.
In numerose applicazioni un solo marker non è sufficiente per definire un sistema di riferimento per visualizzare modelli virtuali all'interno della scena. È necessario, invece, ricorrere ad un insieme di marker, definiti rispetto ad un'origine globale, detto M*arkerfield* (o *marker-set*). In questo modo sarà sempre possibile costruire un sistema di riferimento assoluto, anche in presenza di un marker occluso o non più presente nella scena.
È stata sviluppata un'applicazione di calibrazione che può essere eseguita all'avvio della *fase operatore* con lo scopo di creare un sistema di riferimento assoluto su cui basare il rendering dei modelli 3D durante l'esecuzione on-line del sistema. Il programma di calibrazione deve essere in grado di riconoscere sempre almeno due marker nello stesso istante in modo che sia possibile stabilire relazioni con l'origine del sistema, il *base marker*.
Il processo automatico di creazione del sistema di riferimento assoluto può essere suddiviso in tre sottoprocessi:

1. Individuazione e riconoscimento dei singoli marker
2. Calcolo della matrice di trasformazione tra coppie di marker
3. Calcolo della matrice di trasformazione tra ogni marker e l'origine globale.

Il tracciamento preventivo di un Markerfield, ed il conseguente processo automatico di calibrazione, si rende necessario poiché non sempre è possibile rilevare in modo accurato con strumenti di misura tradizionali la posizione relativa (traslazione e rotazione) nello spazio tra i vari marker[52]. Ad esempio, nel caso in cui i marker siano posizionati molto distanti tra di loro e con angolazioni arbitrarie, sarebbe poco accurato e troppo dispendioso, considerando la richiesta di operare in real-time, effettuare una misura diretta della distanza fra i marker per la calibrazione del sistema, utilizzando, ad esempio, un laser scanner. La calibrazione automatica del Markerfield[71] appare, quindi, un metodo efficiente per rilevare le relazioni spaziali fra i marker, utilizzando gli stessi strumenti hardware e software richiesti durante l'esecuzione dell'applicazione. L'origine del sistema di riferimento assoluto potrà essere un punto all'interno del Markerfield o un marker stesso[7].
Come già detto in precedenza, l'obiettivo che il presente lavoro di tesi si prefigge è quello di sviluppare un sistema di tracking ottico che sia svincolato da geometrie fiduciali e pre-confezionate e si basi, invece, su features naturali presenti

nell'ambiente di lavoro dell'utente. Benché si possano definire più features naturali e sviluppare algoritmi in grado di rilevare la posizione della telecamera in funzione di esse, lo studio del processo di riagganciamento periodico tra features locali è un tema di ricerca ancora aperto e non banale.
È stato sviluppato un algoritmo per la calibrazione del sistema che, per ogni pattern planare, rilevi in modo automatico ciascun marker naturale e ne calcoli la posizione rispetto ad un sistema di riferimento globale, scelto dall'utente al momento dell'esecuzione.
Con questo algoritmo, illustrato qui di seguito, i marker possono essere posti ad una qualsiasi distanza tra di loro, ruotati di angoli arbitrari, non devono essere necessariamente posizionati sul medesimo piano e l'origine globale viene definita rispetto ad un *base marker*.

4.7.2.1. Individuazione e riconoscimento dei marker

In questa fase, l'utente si muove all'interno dell'area del Markerfield per cercare i marker che lo compongono, dove le immagini dei marker da ricercare sono state acquisite in una precedente fase di editing off-line.
Per ogni marker individuato e riconosciuto, viene memorizzato in una specifica struttura l'ID corrispondente, viene calcolata la matrice di trasformazione T_{ID} del marker rispetto alla videocamera ed assegnata ad essa un valore di confidenza c_{ID} che rappresenta la qualità con la quale viene effettuato il riconoscimento del marker o l'accuratezza della matrice di trasformazione *TID*. Questo coefficiente c_{ID} è oggetto del programma di calibrazione per ogni singolo marker.

4.7.2.2. Calcolo della matrice di trasformazione tra coppie di marker

Quando vengono riconosciuti per la prima volta i marker naturali m1 ed m2, viene calcolata la corrispondente matrice di trasformazione T_{m1m2} e cioè:

$$T_{m1m2} = {T_{m1}}^{-1} \cdot T_{m2} \tag{4.1}$$

che fornisce la posizione del marker naturale m2 relativamente al marker naturale m1.
Come è noto, la matrice di trasformazione è una matrice 3x4, le cui prime tre colonne rappresentano la rotazione del marker m2 rispetto al marker m1, mentre la quarta colonna rappresenta la posizione lungo i tre assi di traslazione.
Questa matrice viene memorizzata in memoria dinamica in una struttura che descrive la relazione tra il marker m1 ed il marker m2.
Viene anche calcolato un valore di confidenza c_{m1m2} per la matrice di trasformazione relativa T_{m1m2} che si basa sui singoli valori di confidenza c_{m1}e c_{m2} delle matrici T_{m1}e T_{m2}: una possibile scelta è, ad esempio, la media aritmetica di c_{m1}e c_{m2} oppure il valore minimo tra i due.
Poiché una sola matrice non è sufficiente per fornire una posizione accurata del marker m2 rispetto al marker m1, risulta necessario memorizzare *N* matrici di

trasformazione e applicare un filtro per eliminare quelle che forniscono una posizione che si discosta eccessivamente dal valore reale.
Il filtro viene applicato alle componenti di traslazione della matrice (quarta colonna) e scarta le matrici la cui differenza, rispetto ai valori medi, sia superiore di una certa soglia prefissata. Con le restanti effettua nuovamente una media e calcola la matrice di trasformazione.
L'algoritmo di calibrazione funziona fino a quando le relazioni tra i marker naturali restano statiche, ovvero fino a quando i marker mantengono le stesse posizioni tra di loro ed è possibile vedere contemporaneamente almeno una coppia di marker per permettere il processo di calibrazione.

4.7.2.3. Calcolo della matrice di trasformazione tra ogni marker e l'origine

Dopo aver calcolato le relazioni spaziali di una singola coppia di marker, è possibile procedere con la calibrazione inquadrando una nuova scena e nuovi marker.
Ogni volta che vengono identificati nuovi marker naturali il programma chiede di scegliere rispetto a quale marker, tra quelli presenti nella scena, si vogliano calcolare le matrici di trasformazione.
È possibile, a questo punto, ottenere la matrice di trasformazione tra il base marker ed il singolo marker individuato, moltiplicando le matrici di trasformazione associate al marker scelto come riferimento. La matrice di trasformazione tra il base marker *b* e il marker *m* sarà il prodotto di tutte le matrici di trasformazione presenti tra i due:

$$T_{bm} = T_{bm1} \cdot T_{m1m2} \dots T_{mN\,m} \tag{4.2}$$

dove *m*1…*mN* appartengono al cammino minimo dal base marker al marker m.
Se l'origine del sistema di coordinate coincide con il base marker, la matrice di trasformazione di quest'ultimo sarà una matrice identità.
L'insieme di tutte le matrici di trasformazioni, dal base marker ad ogni marker, definisce il Markerfield, che può essere memorizzato in un file, in modo da poter essere utilizzato durante l'applicazioni di realtà aumentata nel caso di multi-marker tracking.

Capitolo 5
Casi applicativi

Per validare le funzionalità del sistema realizzato e presentato nel capitolo 4 si è scelto di sviluppare alcuni casi applicativi in cui l'operatore viene assistito dal sistema nello svolgimento di attività di manutenzione. I *test* sono stati realizzati presso ITIA-CNR con la collaborazione di aziende partner, quali Afros S.p.A.
Ogni caso applicativo ha utilizzato entrambe le fasi di esecuzione del sistema (*fase istruttore* e *fase operatore*, vedi capitolo 4). Per ogni caso applicativo si è analizzata la sequenza di operazioni da compiere, la si è tradotta in operazioni di controllo per il sistema ed è stata creata la relativa procedura usando l'interfaccia grafica dell'applicazione ***istruttore.exe***. Il file con la procedura così salvato viene caricato durante l'esecuzione del programma ***operatore.exe***.
Il sistema applicato ai *test cases* risulta autonomo nell'esecuzione della *fase operatore*: una volta riconosciuto il marker naturale all'interno dell'ambiente, assiste l'utente mostrandogli le operazioni da compiere, controlla che le operazioni compiute dall'utente siano coerenti con il flusso logico della procedura scelta tra quelle memorizzate nel database, lo avvisa in caso di errata esecuzione ed avanza autonomamente al passo successivo.

5.1. Preparazione dei *casi applicativi*

L'hardware impiegato nei casi applicativi è composto da:

- notebook con processore Intel Core 3 2,27 GHz, Memoria Ram da 4GB e scheda video ATI Radeon HD 5470 con 1 Gb di memoria dedicato. Sistema operativo utilizzato: Windows 7 Home Premium.
- una webcam Logitech HD Pro C920[4] collegata al pc e montata sulla testa dell'operatore. I parametri di configurazione sono stati impostati sulla gestione manuale del contrasto e della messa a fuoco in modo da evitare anche eventuali adattamenti automatici del driver della webcam a fronte di variazioni di luminosità ambientale che potrebbero interferire con gli algoritmi di computer vision. La calibrazione della telecamera è gestita nella fase istruttore (vedi capitolo 4) e viene eseguita una sola volta.
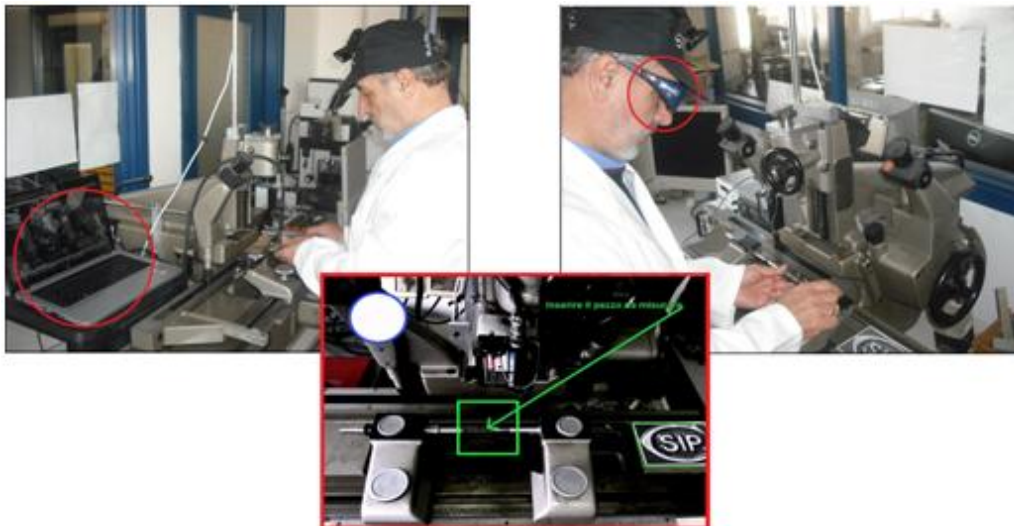
Figura 5.1 – Esempio di configurazione hardware durante l'esecuzione. A sinistra, l'operatore visualizza le informazioni su di un monitor per PC. A destra, l'operatore indossa un HMD. In basso, particolare della schermata dell'interfaccia utente visualizzata in tempo reale.

All'interno dell'ambiente di lavoro non viene posto nessun altro marker, in aggiunta a quelli selezionati dall'utente, e non vengono utilizzati sistemi di illuminazione aggiuntivi in quanto gli algoritmi scelti ed utilizzati (identificazione features e riconoscimento ed inseguimento di oggetti) sono robusti alla variazione di illuminazione e non necessitano di particolari condizioni di luce se non quella già presente all'interno di un normale ambiente di lavoro industriale.

[4] http://www.logitech.com/it-it/webcam-communications/webcams

5.2. Caso 1 – Sostituzione della scheda RAM in un PC desktop

5.2.1. Fase istruttore: Creazione della procedura

Le operazioni che l'operatore deve compiere per eseguire il proprio compito devono essere descritte in modo tale che il sistema possa interpretarne facilmente la corretta esecuzione, utilizzando le funzionalità presentate nel capitolo 4.
La sequenza operativa di questo caso applicativo consiste nella rimozione del coperchio del case del PC, nell'identificazione e successiva rimozione dell'hard disk per poter rendere visibile lo slot di inserimento della scheda di memoria RAM e quindi nella sostituzione della scheda stessa. Prima di procedere al suo inserimento, però, l'utente dovrà scegliere fra più alternative quale scheda inserire nell'apposito slot. Il sistema riconosce e valuta le decisioni dell'utente per avanzare nella procedura e mostrargli contestualmente le informazioni necessarie. Dopo aver inserito la scheda RAM selezionata si procede all'assemblaggio finale: si posiziona l'hard disk e si richiude il case del PC. La sequenza di operazioni identificata è schematizzata in **Tabella 5.1**.

Operazione manuale	Descrizione dell'operazione	Definizione del controllo operativo del sistema	Codice DSS
Apertura del case	Sollevamento del coperchio del case e sua rimozione dalla scena	Ricerca dell'etichetta e suo inseguimento fino a coordinate definite dal marker	Follow
Estrazione dell'hard disk	Identificazione e rimozione dell'hard disk	Ricerca delle coordinate del pezzo e verifica della sua presenza	BlobPG
Selezione della scheda RAM	Selezione della scheda corretta tra diversi componenti	Riconoscimento di due componenti diversi ed interazione con le scelte dell'utente	TMatch
Inserimento della scheda RAM	Inserimento della scheda selezionata all'interno dello slot	Ricerca delle coordinate dello slot e verifica della presenza della scheda	BlobPG
Posizionamento dell'hard disk	Inserimento dell'hard disk nella propria sede	Ricerca delle coordinate in cui dovrà trovarsi il pezzo	BlobPG
Chiusura del case	Posizionamento del coperchio del case	Rilevamento della presenza dell'etichetta del case	PosE

Tabella 5.1 – Codifica delle operazioni: dal manuale di istruzioni alla realtà aumentata.

Nella colonna *Operazione Manuale* è indicata la descrizione sintetica dell'operazione che l'utente deve compiere. Nella colonna *Descrizione dell'operazione* il singolo passo è analizzato con maggiore dettaglio e nella colonna *Definizione del Controllo Operativo del Sistema* è tradotto in una serie di operazioni primitive che il sistema deve eseguire per operare il controllo sulle azioni dell'utente. Nella colonna *Codice DSS* è indicato, infine, il codice sintetico riferito alla funzione del modulo di visione artificiale (capitolo 4) che si utilizza nel passo operativo.
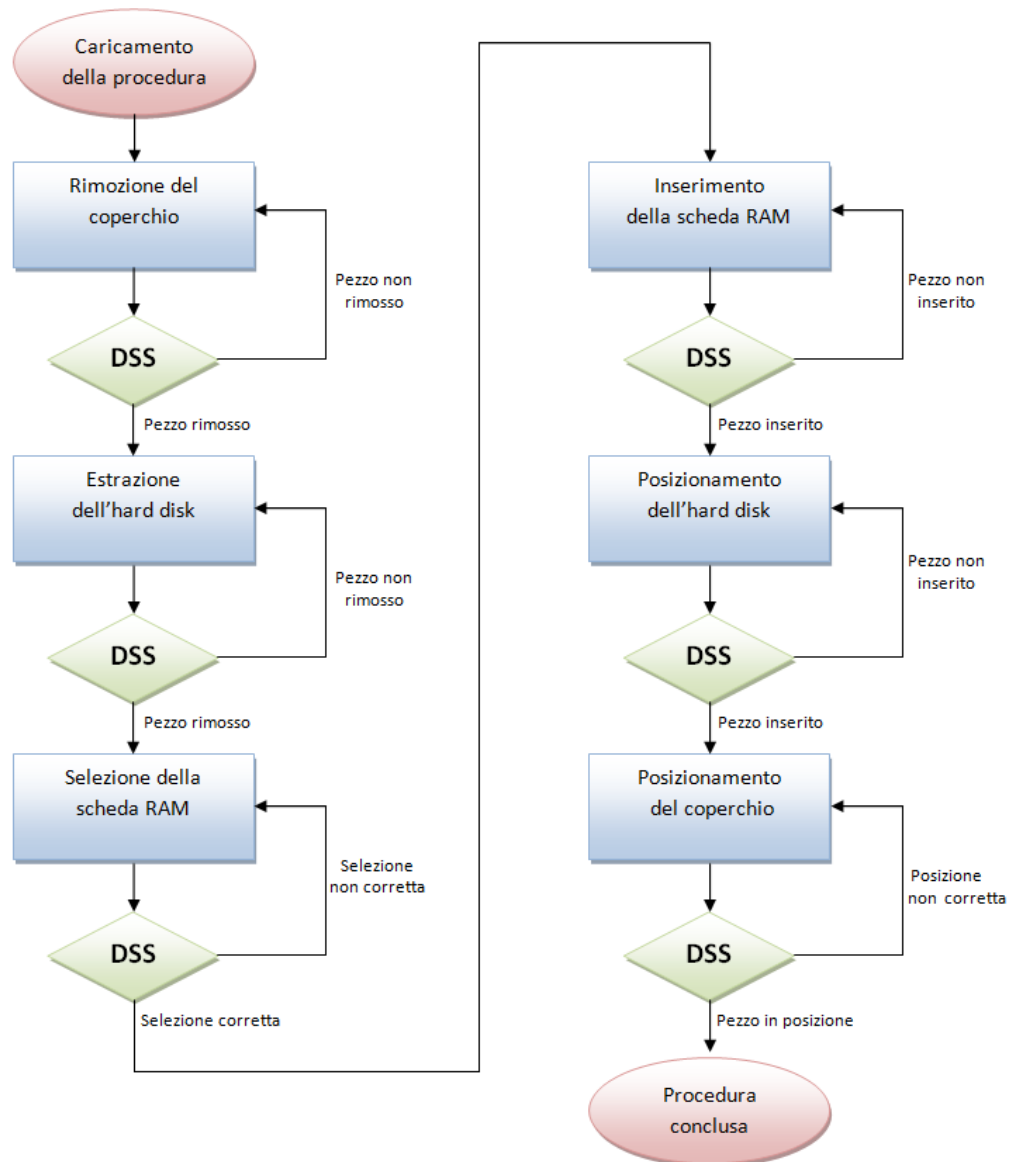
Figura 5.2 – Diagramma di flusso della procedura del caso applicativo: il modulo di supporto decisionale (DSS) legge le istruzioni contenute nella procedura ed interroga il modulo di visione (CVM) per controllare lo stato di avanzamento delle operazioni dell'utente.

Prima di procedere alla definizione dei parametri di input e di verifica dei singoli passi operativi, si possono schematizzare le regole di funzionamento in un diagramma di flusso(**Fig. 5.2**).
Ogni rilevamento dello stato del sistema e dei riferimenti (marker naturali) è effettuato dal sistema di visione e controllato dal modulo decisionale (DSS) che indica al sistema come procedere, eseguendo le istruzioni codificate nella procedura. Dopo aver letto le informazioni contenute nel passo operativo, il DSS attiva secondo necessità il modulo di realtà aumentata (ARM) per eseguire il rendering di modelli virtuali o descrizioni testuali, passando in ingresso anche le coordinate del marker rilevato dal CVM. A seconda degli stati possibili del sistema, descritti nel passo operativo di riferimento, il DSS avanza nella procedura o apporta azioni correttive alle operazioni dell'utente (**Fig.5.2**).
L'ultimo passo della fase istruttore è la redazione della procedura. Attraverso lo studio preliminare del problema, di cui si è appena discusso, e l'utilizzo dell'interfaccia utente di istruttore.exe, si giunge al salvataggio nel database delle regole e passi operativi che il sistema dovrà usare nello svolgimento della *fase operatore* (**Fig. 5.3**).

	Tipo	Descrizione	Operazione	Ref Marker	Input	Check	Verifica	CondizioneOK
1	☐	Aprire il case	Follow	./PC/PC_marker_0.png	100 -100	100 -300	☐	☐
2	☐	Rimuovere l'Hard Disk	BlobPG	./PC/PC_marker_1.png ./PC/PC_n	1 1	-100 -50 150	☐	☐
3	☐	Selezionare la scheda RAM	Tmatch	0	./PC/3/PC_template_0	40 440	☑	☐
4	☑	Scheda da 512 Mb	Tmatch	-1	-1	1	☐	☐
5	☑	Scheda da 1 Gb	Tmatch	-1	-1	7	☐	☑
6	☐	Inserire la scheda RAM	BlobPG	./PC/PC_marker_1.png ./PC/PC_m	1 1	-50 -50 150	☐	☐
7	☐	Inserire l'Hard Disk	BlobPG	-1	1 2	-50 -50 150	☐	☐
8	☐	Chiudere il case	PosE	./PC/PC_marker_3.png	320 240 330	0.05	☐	☐

Figura 5.3 – Procedura del caso applicativo 1.

5.2.2. Fase operatore: Esecuzione della procedura

Il sistema in modo autonomo esegue la procedura (**Fig. 5.3**) ed interpreta le modifiche apportate dall'utente per prendere decisioni e modificare di conseguenza il flusso logico del processo schematizzato nel paragrafo precedente.
Come descritto nel capitolo 4, ogni passo operativo è un insieme di dati che permette al sistema il controllo dell'esecuzione di una singola operazione. La procedura è un insieme di passi operativi che il sistema esegue in modo ordinato e sequenziale.
Con riferimento alla **Fig. 5.3**, il campo *Descrizione* contiene il testo da visualizzare sul display dell'operatore, nel campo *Operazione* è codificato il nome della funzione di computer vision che il sistema utilizza per esercitare il controllo sulle azioni dell'utente. I campi *Input* e *Check* contengono, rispettivamente, i parametri in ingresso e di controllo dell'operazione identificata nell'omonimo campo. Nel campo *Ref Marker* è presente il percorso del file immagine relativo ai marker naturali che si utilizzeranno come riferimento per l'operazione corrente. La casella *Verifica* è spuntata se

l'operazione richiede una verifica aggiuntiva. Le righe contrassegnate dalla selezione del campo *Tipo* sono, appunto, verifiche aggiuntive o possibili errori. Tra tutti, quello che presenta la *CondizioneOK* con la spunta è lo stato di corretta esecuzione.

Passo 1. Rimozione del coperchio

Il sistema di visione individua il marker selezionato (**Fig.5.4**), utilizzando l'algoritmo di tracciamento (capitolo 4), e, a partire dalla sua posizione nello spazio, identifica il coperchio del case, definendone un template di coordinate definite in input, applicando la funzione `Follow` del modulo di Visione artificiale (CVM). Il modulo di realtà aumentata (ARM), utilizzando le coordinate di check, renderizza un rettangolo che identifica la posizione in cui il template del coperchio dovrà trovarsi per proseguire con il passo operativo successivo (**Fig.5.5**). L'utente solleva il coperchio ed il CVM ne valuta in tempo reale la posizione fino al raggiungimento dell'obiettivo. Quando le coordinate immagine del template restituite dal CVM al DSS rispettano i vincoli definiti nel passo operativo il DSS avanza al passo operativo successivo (**Fig. 5.6**).

Passo 2. Estrazione dell'hard disk

Poiché la riuscita di questa operazione dipende fortemente dal tracking del marker naturale, si è scelto di salvare più immagini dello stesso marker in differenti condizioni ambientali, per garantirne sempre il riconoscimento (**Fig. 5.7**).
Nella casella di check viene chiesto al sistema di osservare i blob presenti all'interno di un'area dell'immagine, colorata in verde, le cui coordinate sono relative al marker individuato (**Fig. 5.8**). I parametri in input permettono di gestire le dimensione (area e rapporto di forma) del blob che il CVM deve cercare applicando la funzione `BlobPG`. Il DSS aspetta che il CVM rilevi l'effettiva rimozione dell'hard disk, confrontando due immagini successive (paragrafo 4.2.2), istruisce l'ARM per informare l'utente ed avanza al passo operativo seguente.

Passo 3. Selezione della scheda RAM

Questa operazione mette in rilievo la capacità di interazione del sistema con l'operatore. Nella fase istruttore si insegna al sistema a riconoscere due differenti schede utilizzando l'algoritmo di pattern matching DOT (capitolo 1), salvate nel file template in input. In questa operazione non viene selezionato alcun marker naturale. Si identifica, invece, un'area del display che funge da interfaccia utente. L'operatore mostra al sistema i differenti componenti e riceve informazioni testuali sulle loro caratteristiche. Il CVM attraverso la funzione `TMatch` identifica i diversi oggetti restituendo l'indice numerico del template individuato (**Fig 5.10**). Il DSS confronta tali valori con quelli definiti nella casella check del passo operativo, legge le informazioni relative al template corrispondente e le trasmette all'ARM per la visualizzazione sul display.
Per concludere l'operazione, l'utente deve selezionare una scheda trascinandola nell'area dell'immagine evidenziata dal cerchio rosso in basso a sinistra (**Fig.9**).
Il sistema conferma la selezione variando il colore (da rosso a verde) dell'area di controllo in cui l'utente sposta il pezzo (**Fig.11**).

Figura 5.5 – Marker naturale da riconoscere.
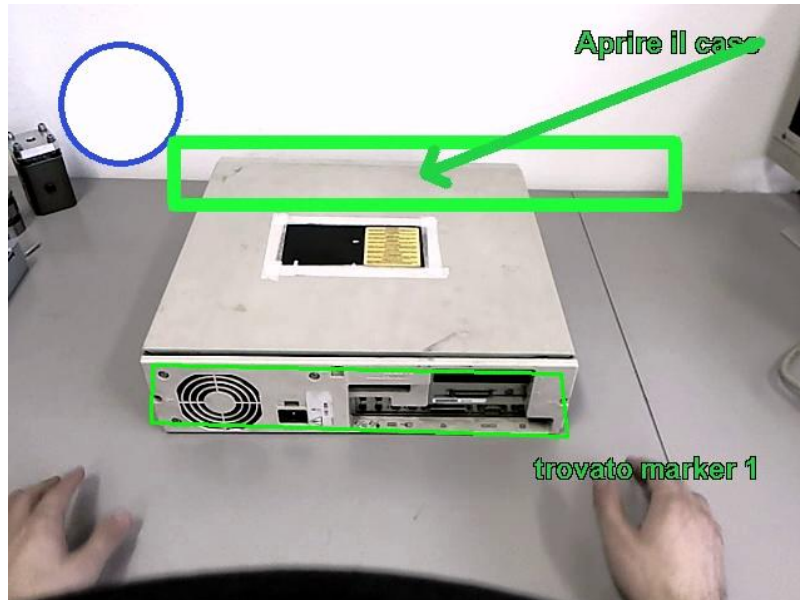
Figura 5.4 – Ricerca del marker e visualizzazione delle istruzioni.
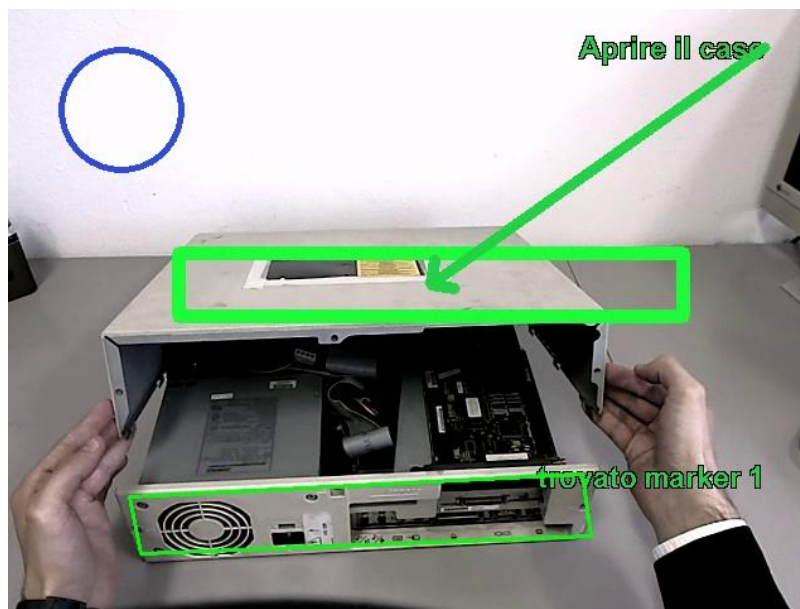
Figura 5.6 – Esecuzione dell'operazione fino alla sua verifica.
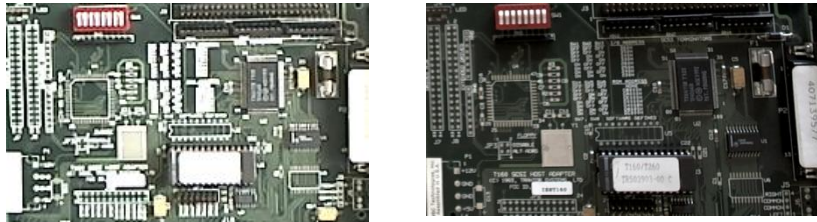
Figura 5.8 – Marker naturale in differenti condizioni di illuminazione.

Figura 5.7 – Rilevamento del marker e visualizzazione del rettangolo di ricerca.

Figura 5.9 – Identificazione di una scheda e visualizzazione di informazioni testuali aggiuntive (in rosso).
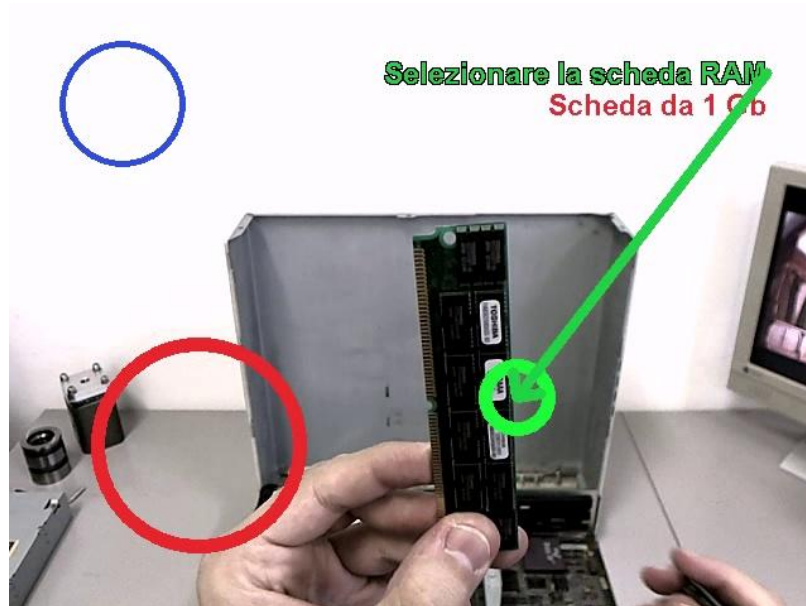
Figura 5.10 – Identificazione di una scheda differente.
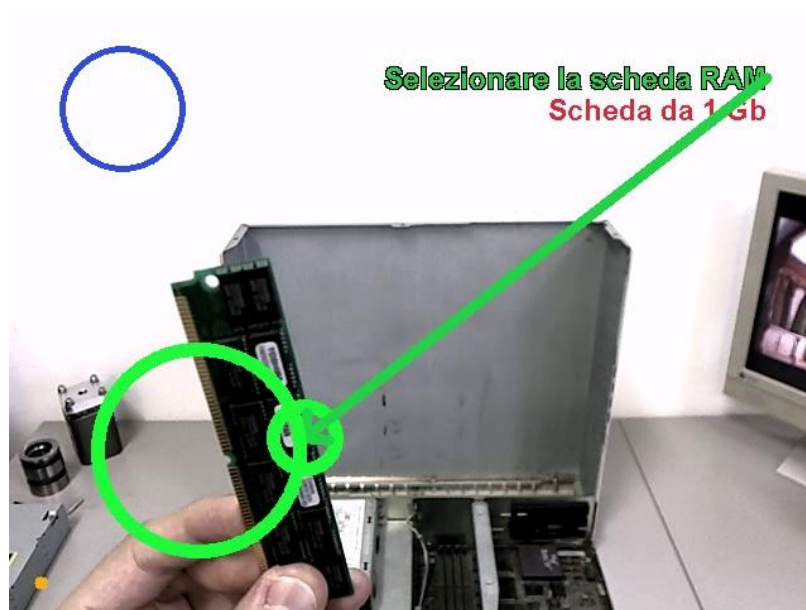
Figura 5.11 – Conferma della selezione.
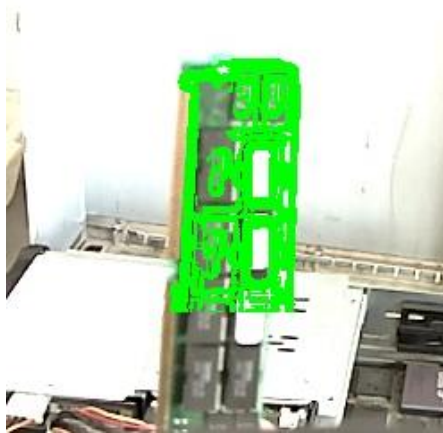
Figura 5.12 – Esecuzione dell'operazione TMatch.

Passo 4. Inserimento della scheda nell'apposito slot

La scheda selezionata al passo operativo precedente viene inserita dall'utente nell'apposito slot (**Fig. 5.13**). Il sistema monitora questa operazione, come per il Passo 2, applicando la funzione `BlobPG` del CVM che individua nel flusso video differenza i blob che soddisfano i vincoli di forma posti in input e ne restituisce le coordinate spaziali. Il DSS interpreta l'output del CVM e li confronta con le condizioni di verifica: se uno dei blob individuati è all'interno di un'area di verifica, l'operazione è conclusa.
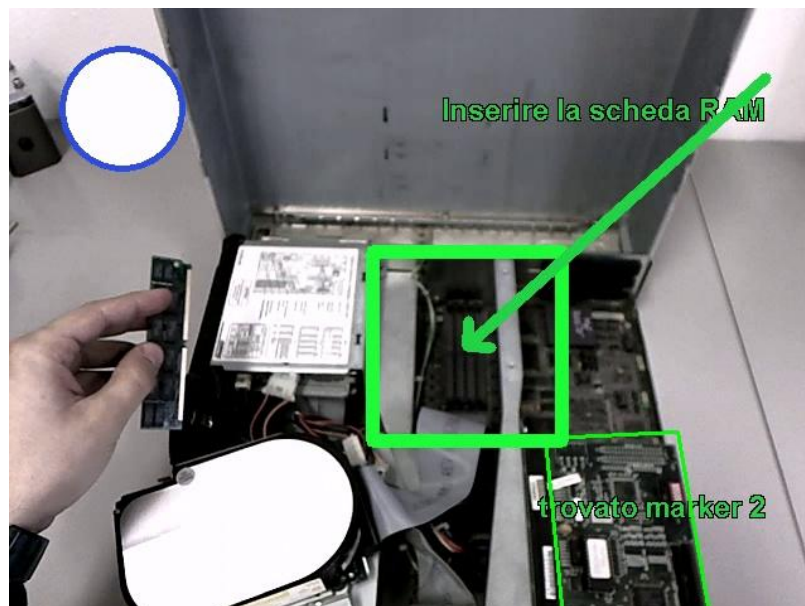
Figura 5.13 – Verifica dell'inserimento della scheda RAM.

Passo 5. Inserimento dell'hard disk

La procedura prevede che l'hard disk estratto al Passo 2 sia collocato nuovamente nella sua sede. Il controllo dell'operazione è molto simile al precedente (**Fig. 5.14**).
I parametri di controllo che il DSS legge dal passo operativo e comunica al CVM, però, consentono la ricerca di un blob di dimensioni maggiori rispetto a quello che caratterizza la scheda RAM (**Fig. 5.15**).
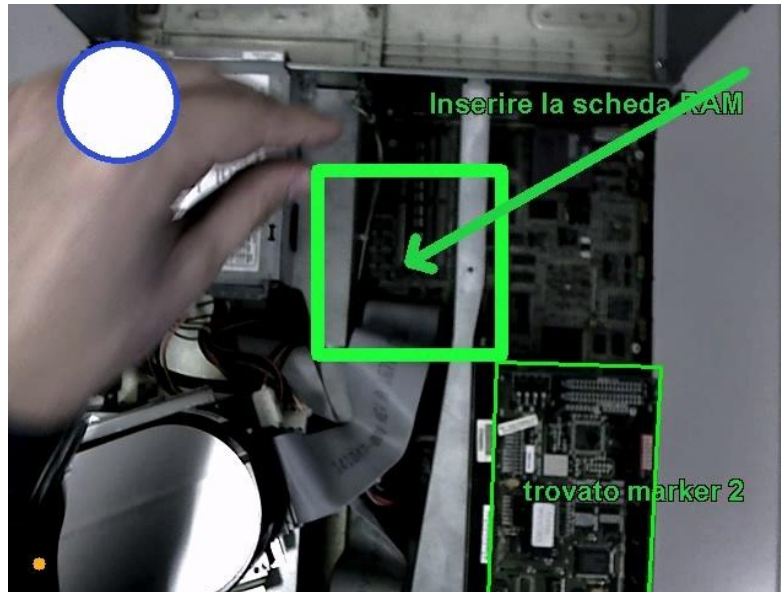
Figura 5.14 – Controllo dell'esecuzione. Le coordinate della regione in verde dipendono dal tracciamento real-time del marker.
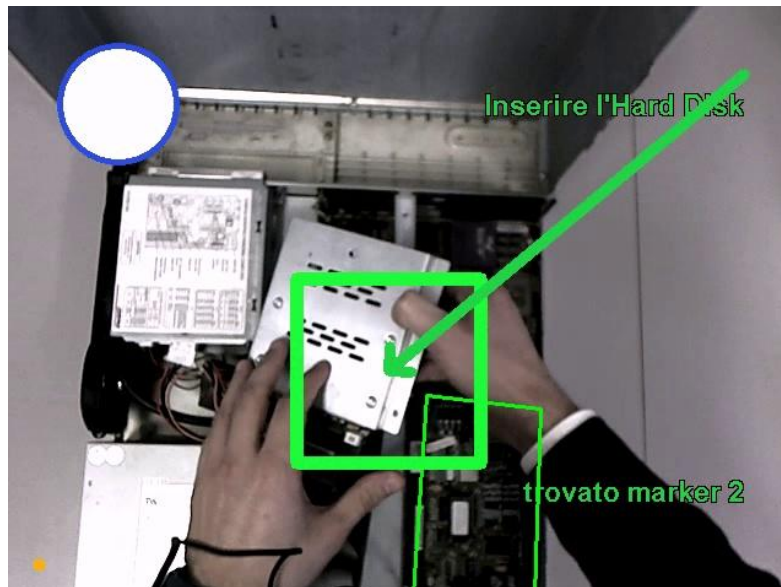
Figura 5.15 – Il filtro sulle dimensioni dei blob serve per evitare falsi positivi a causa di blob troppo piccoli (movimento della telecamera) o troppo grandi (mani dell'operatore).

Passo 6. Posizionamento del coperchio

Per concludere la procedura il case deve essere chiuso. Il sistema di visione cerca il marker naturale selezionato (in questo caso l'etichetta sul coperchio) e arresta l'esecuzione solo quando il riconoscimento viene considerato stabile, cioè quando la differenza fra i valori di rototraslazione calcolati per due acquisizioni successive è inferiore alla soglia di verifica. Il DSS trasmette all'ARM le coordinate del marker, per visualizzare un rettangolo verde intorno all'oggetto ed il messaggio di testo relativo alla corretta conclusione della procedura.
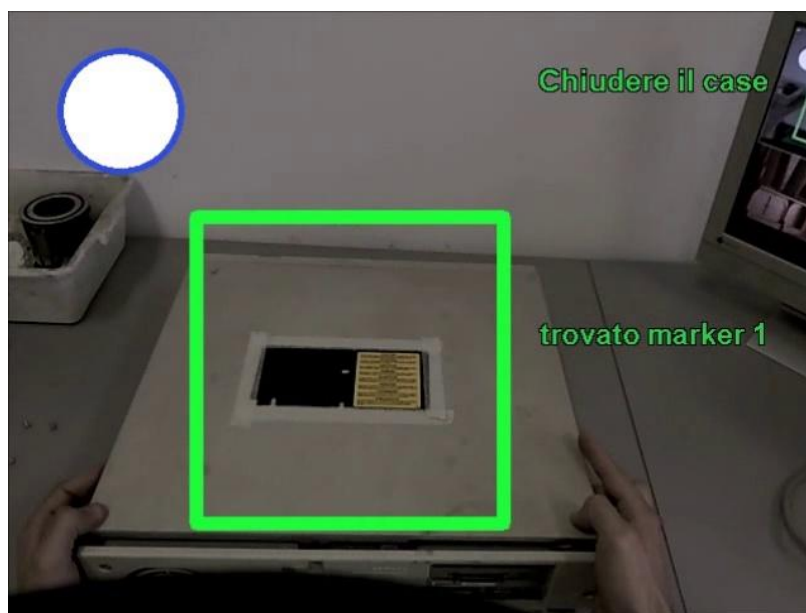
Figura 5.16 – Il coperchio è nella posizione corretta se viene riconosciuto il marker naturale.

5.3. Caso 2 – Sostituzione della membrana di tenuta in una valvola on/off

5.3.1. Fase istruttore: Creazione della procedura

Seguendo la stessa metodologia presentata nel paragrafo 5.2, l'utente istruttore deve redigere la procedura che il sistema dovrà eseguire e monitorare partendo da un'attenta analisi della sequenza dei compiti che l'operatore deve compiere. Partendo dalla sequenza di azioni elementari, incrementando ad ogni passo il livello di dettaglio, si arriva a definire un elenco di operazioni primitive di computer vision che il sistema possa gestire.

Il presente caso applicativo è stato condotto su di un componente meccanico, prodotto dall'azienda Afros s.p.a. L'azienda, situata nei pressi di Milano, è il principale fornitore mondiale di tecnologie per la lavorazione di poliuretani. Afros progetta, costruisce e commercializza in tutto il mondo la più ampia gamma di macchinari, dalle singole teste di miscelazione fino ad impianti completi impiegati nell'ambito delle principali produzioni industriali.

L'oggetto dello studio è una *valvola on/off* che regola il flusso di mandata dell'olio in pressione all'interno di una testa di miscelazione per la produzione del poliuretano (**Fig. 17**).
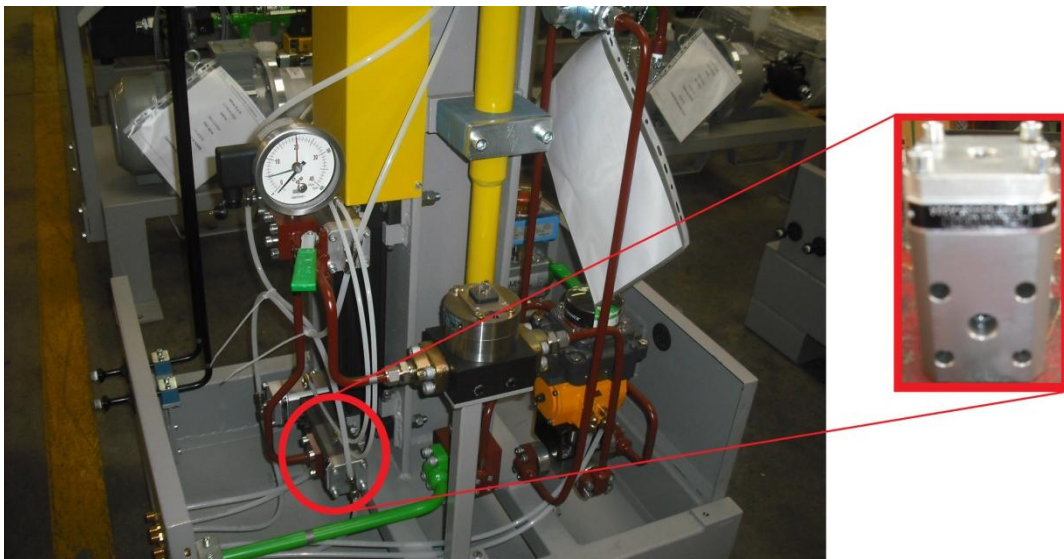
Figura 5.17 – Valvola on/off e sua collocazione sulla macchina.

Le operazioni da compiere per condurre la manutenzione del componente sono quelle di rimozione del coperchio della valvola, svitando le 4 viti di collegamento, di sostituzione della guarnizione e di ri-assemblaggio del coperchio e delle viti.

La guarnizione di tenuta è montata sullo spintore, un perno che aziona le valvole di mandata dell'olio, azionato da un secondo condotto oleodinamico.

La sostituzione si può scomporre in due fasi, la selezione della guarnizione adatta, che avviene considerando le caratteristiche meccaniche dei materiali con cui si realizzano tali guarnizioni, e il disassemblaggio del componente spintore, a sua volta costituito da piccoli particolari meccanici. Si è scelto di non monitorare quest'ultima operazione a causa delle ridotte dimensioni dei componenti e della specificità delle tolleranze dimensionali dell'operazione che non possono essere monitorate dal sistema di visione scelto nel presente lavoro: dopo aver rimosso lo spintore e selezionato la guarnizione, il sistema attenderà che l'utente monti la nuova guarnizione e monitorerà, poi, che lo spintore sia montato correttamente.

Operazione manuale	Descrizione dell'operazione	Definizione del controllo operativo del sistema	Codice DSS
Rimozione delle viti 1–2–3	Svitare e rimuovere la vite indicata	Identificare le coordinate della vite e verificarne la rimozione	BlobPG
Rimozione della vite 4 e del coperchio	Svitare la vite indicata e rimuovere il coperchio della valvola	Identificare le coordinate della vite e verificare la rimozione del componente	BlobPG
Estrazione dello spintore	Afferrare lo spintore ed estrarlo dalla propria sede	Riconoscere la presenza del pezzo nella scena e verificare la sua posizione all'interno dell'immagine	TMatch
Selezione della guarnizione corretta	Identificare la guarnizione corretta tra differenti alternative	Identificare componenti diversi e verificare la selezione dell'utente	RGBlob
Sostituzione della guarnizione	Sostituire la guarnizione selezionata	Attendere che l'utente effettui la manutenzione del componente	Attesa
Inserimento dello spintore	Inserire lo spintore nella propria sede	Riconoscere la presenza del pezzo nella scena e verificare la sua posizione all'interno dell'immagine	TMatch
Posizionamento del coperchio e della vite 4	Posizionare il coperchio sulla valvola e inserire la vite selezionata	Identificare le coordinate in cui verificare la presenza del pezzo	BlobPG
Inserimento delle viti 3–2–1	Inserire o avvitare la vite selezionata	Identificare le coordinate in cui verificare la presenza del pezzo	BlobPG

Tabella 5.2 – Codifica delle operazioni: dal manuale di istruzioni alla realtà aumentata.

La sequenza di operazioni identificata è schematizzata in **Tabella 5.2**.
Nella colonna *Operazione Manuale* è indicata la descrizione sintetica dell'operazione che l'utente deve compiere. Nella colonna *Descrizione dell'operazione* il singolo passo è analizzato con maggiore dettaglio e nella colonna *Definizione del Controllo Operativo del Sistema* è tradotto in una serie di operazioni primitive che il sistema deve eseguire per operare il controllo sulle azioni dell'utente. Nella colonna *Codice DSS* è indicato, infine, il codice sintetico riferito alla funzione del modulo di visione artificiale (capitolo 4) che si utilizza nel passo operativo.
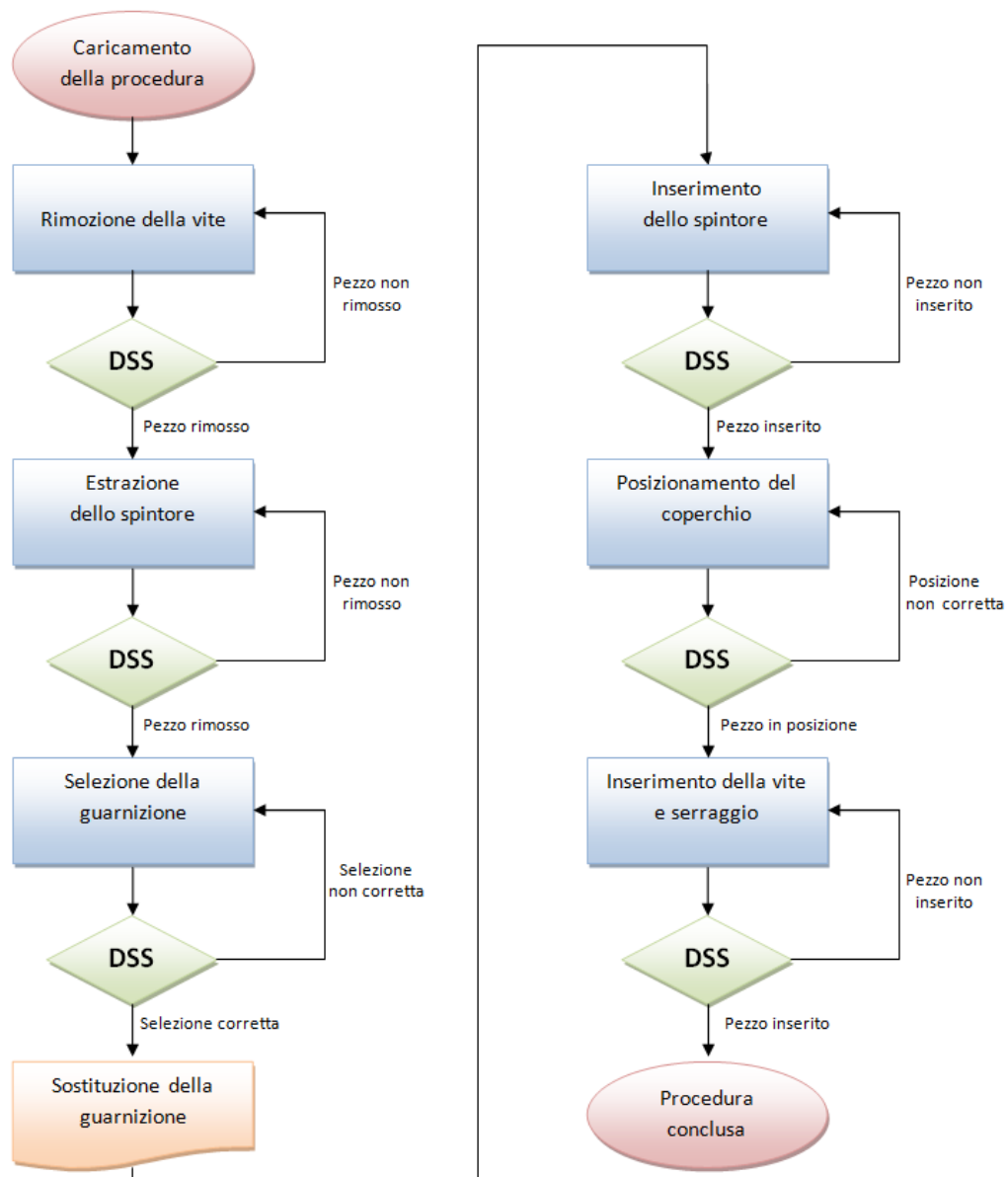
Figura 5.18 – Diagramma di flusso della procedura del caso applicativo 2.

Il diagramma di flusso che rappresenta le regole di funzionamento seguite nella progettazione della procedura è rappresentato in **Fig. 5.18**. Ogni rilevamento dello stato del sistema e dei riferimenti (marker) naturali è effettuato dal sistema di visione e controllato dal modulo decisionale (DSS) che indica al sistema come procedere.
Il DSS, infatti, legge le istruzioni contenute nella procedura ed interroga il modulo di visione (CVM) per controllare lo stato di avanzamento delle operazioni dell'utente. Sulla base della risposta del CVM restituisce un feedback visuale all'utente sfruttando le funzionalità del modulo di realtà aumentata (ARM).
L'ultimo passo della fase istruttore è la redazione della procedura. Attraverso lo studio preliminare del problema, di cui si è appena discusso, e l'utilizzo dell'interfaccia utente dell'applicazione istruttore.exe, si giunge al salvataggio nel database delle regole che il sistema dovrà seguire nello svolgimento della *fase operatore* (**Fig. 5.19**).

Indice	Tipo	Descrizione	Operazione	Ref Marker	Input	Check	Modelli3D	path	PosRel	Verifica	Condizione(
1	☐	Rimuovere la vite 1	BlobPG	./AFROS/AFROS_marker_2.png ./AFROS/AFROS_marker_3.png	1 1	0 -15 50	vite	./modelli3D/vite6x30.3ds	0	☐	☐
2	☐	Rimuovere la vite 2	BlobPG	-1	1 1	45 -35 50	vite	./modelli3D/vite6x30.3ds	0	☐	☐
3	☐	Rimuovere la vite 3	BlobPG	-1	1 1	45 -15 50	vite	./modelli3D/vite6x30.3ds	0	☐	☐
4	☐	Rimuovere il coperchio	BlobPG	-1	1 1	5 -30 50	vite	./modelli3D/vite6x30.3ds	0	☑	☐
5	☑	e la vite 4	BlobPG	-1	0.5 2	20 -15 200				☐	☑
6	☐	Estrarre lo SPINTORE	Tmatch	./AFROS/AFROS_marker_0.png ./AFROS/AFROS_marker_1.png	./AFROS	1 10 10 300 1 1				☐	☐
7	☐	Sostituire la guarnizion	WaitOP	-1	1	100				☐	☐
8	☐	Inserire il nuovo SPINT(	Tmatch	-1	./AFROS	1 10 10 300 0 0				☐	☐
9	☐	Inserire la vite 4	BlobPG	-1	1 1	5 -50 50	vite	./modelli3D/vite6x30.3ds	0	☑	☐
10	☑	dopo aver posizionato i	BlobPG	-1	0.5 2	20 -15 200				☐	☑
11	☐	Inserire la vite 3	BlobPG	./AFROS/AFROS_marker_2.png ./AFROS/AFROS_marker_3.png	1 1	50 -35 50	vite	./modelli3D/vite6x30.3ds	0	☐	☐
12	☐	Inserire la vite 2	BlobPG	-1	1 1	45 -50 50	vite	./modelli3D/vite6x30.3ds	0	☐	☐
13	☐	Inserire la vite 1	BlobPG	-1	1 1	0 -30 50	vite	./modelli3D/vite6x30.3ds	0	☐	☐
14	☐	Serrare le viti con una c	WaitOP	-1	1	100				☐	☐

Figura 5.19 – Procedura del caso applicativo 2.

5.3.2. Fase operatore: Esecuzione della procedura

Il sistema in modo autonomo esegue la procedura (**Fig. 5.19**) ed interpreta le modifiche apportate dall'utente allo stato del sistema per prendere decisioni e modificare di conseguenza il flusso logico del processo schematizzato nel paragrafo precedente.
Senza soffermarsi nuovamente sulla struttura della procedura e sulla definizione dei singoli parametri (per i quali si rimanda al paragrafo 5.2.2 ed alla **Fig. 5.19**), che saranno esplicitati comunque nel seguito, si vuole evidenziare ora una nuova funzionalità del sistema, che è l'inserimento di modelli 3D nella procedura in esame. I modelli 3D (in questo caso delle viti) sono visualizzati nell'interfaccia utente per migliorare la comunicazione uomo-macchina e consentire una più immediata comprensione del compito che l'operatore deve eseguire.

Passo 1. Rimozione delle viti e del coperchio

Il sistema individua il marker naturale selezionato e, a partire dalla sua posizione nello spazio, identifica il coperchio e le viti da rimuovere. Per queste operazioni sono stati utilizzati due marker di riferimento. Il sistema di tracciamento, descritto nel capitolo 4 e sviluppato nel CVM, ne cerca uno ed utilizza l'altro nel caso in cui il precedente non sia più disponibile(**Fig. 5.21**). Il DSS interpreta le coordinate del marker e le invia all'ARM per visualizzare sul display l'area in cui l'utente dovrà operare.
Nella casella di Check della procedura (**Fig. 5.20**) viene chiesto al sistema di osservare i blob presenti all'interno di un'area dell'immagine, colorata in verde, le cui coordinate sono relative al marker individuato. I parametri in Input permettono al CVM di gestire le dimensione (area e rapporto di forma) del blob da ricercare, applicando la funzione `BlobPG`. Quando il sistema di visione riconosce l'effettiva rimozione della vite in oggetto, confrontando due immagini successive (come spiegato nel paragrafo 4.2.2) comunica la riuscita dell'operazione al DSS che a sua volta informa l'utente, attraverso il modulo AR, ed avanza al passo operativo seguente. Durante l'esecuzione dell'applicazione operatore.exe il modello 3D della vite viene visualizzato in alto a sinistra per focalizzare l'attenzione dell'utente sul componente da monitorare. Il DSS trasmette, come indicato nel paragrafo 4.3.1, all'ARM il percorso, le coordinate e le trasformazioni da applicare al modello da caricare e renderizzare.
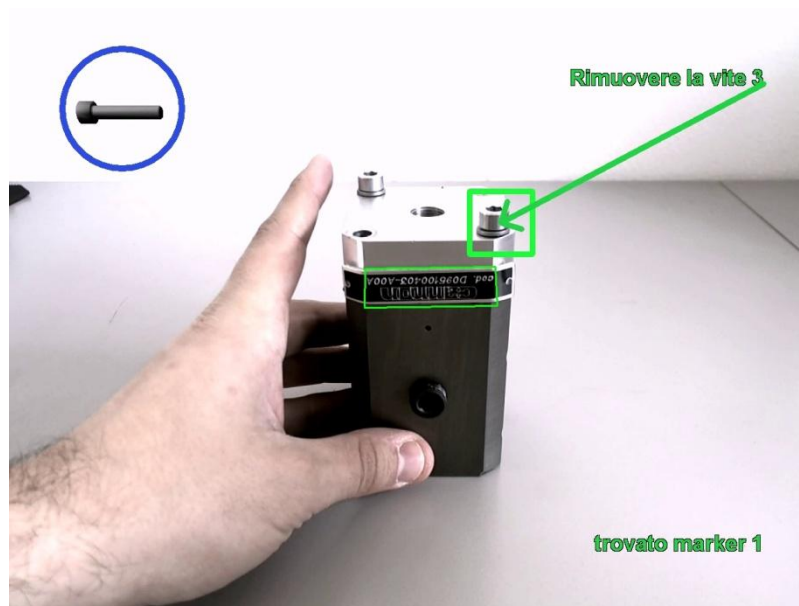
Figura 5.20 – Esempio di esecuzione del passo operativo corrente.

La rimozione dell'ultima vite (la quarta) avviene contestualmente con il coperchio (**Fig. 5.22**). In questo caso il sistema, preventivamente programmato nella fase istruttore (riga 4 di **Fig. 5.19**), cerca un blob di dimensioni maggiori, associato al coperchio, per valutare lo stato di avanzamento dell'operazione.
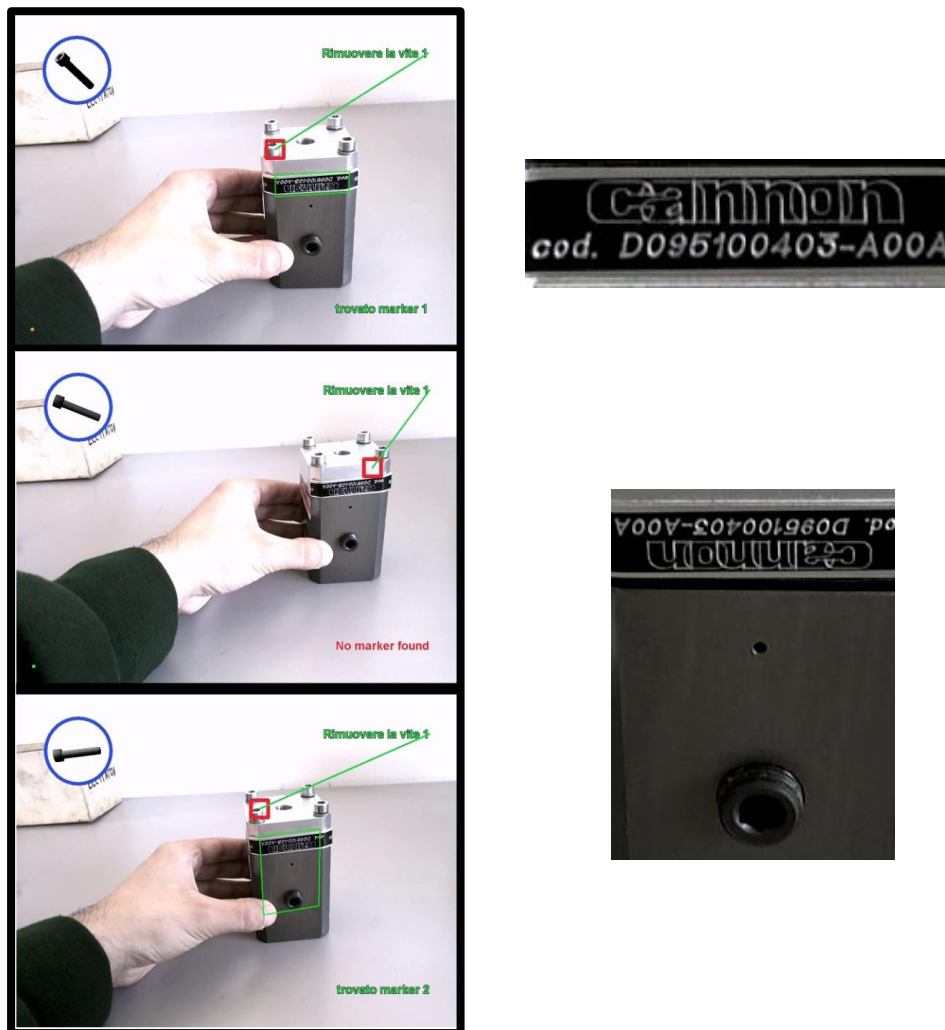
Figura 5.21 – (A sinistra) Sequenza di riagganciamento ad un altro marker: il sistema rileva il marker 1, poi perde il riferimento a seguito di un brusco movimento e rileva il secondo marker per continuare l'esecuzione. (A destra) Marker naturali di riferimento per il passo operativo corrente.
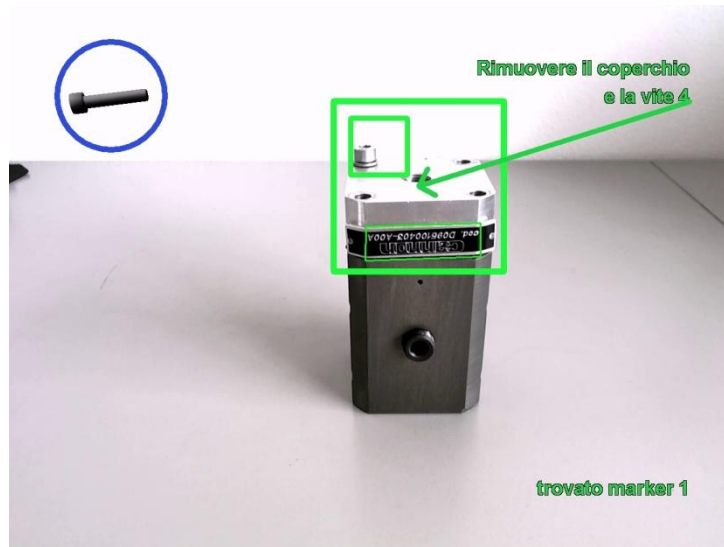
Figura 5.22 – Verifica aggiuntiva: rimozione del coperchio.

Passo 2. Estrazione dello spintore

Il template dello spintore viene registrato durante la *fase istruttore*, con le modalità descritte nel paragrafo 4.6.1.2, ed il percorso del file da caricare è inserito nella casella di input per questa operazione. All'avvio del passo operativo il DSS attiva la funzione `TMatch` del CVM per cercare lo spintore, che si troverà nella propria sede all'interno della valvola. Il sistema registra la posizione iniziale dell'oggetto e, trasmettendo le sue coordinate all'ARM, renderizza un cerchio rosso intorno a tale posizione (**Fig. 5.23**).
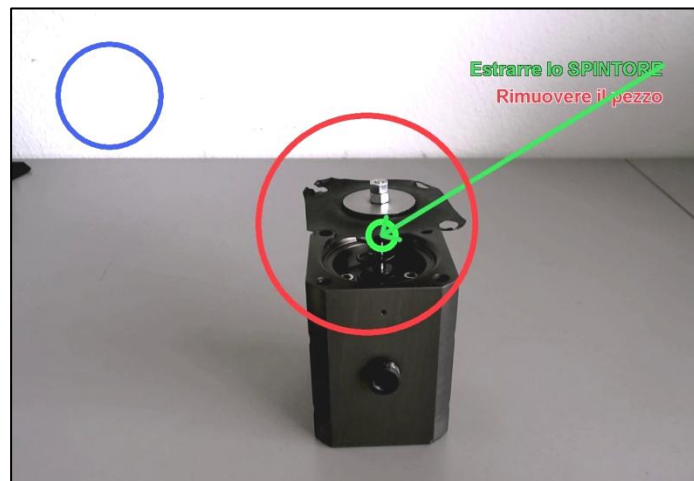
Figura 5.23 – Esecuzione del passo operativo. Lo spintore deve essere rimosso dalla propria sede.

Il DSS confronta le coordinate del template restituite dal CVM con quelle possibili, indicate nel passo operativo, e comunica all'ARM quali istruzioni visualizzare.

Poiché l'operazione risulta correttamente eseguita se lo spintore viene rimosso dalla sua posizione originaria ed estratto dalla valvola, il cerchio diventa verde se l'operatore compie l'operazione corretta (**Fig. 5.25**), altrimenti resta rosso.
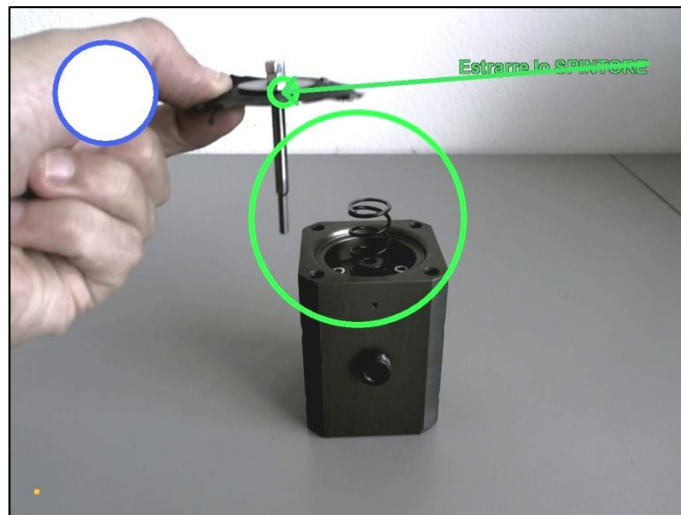
Figura 5.25 – Operazione eseguita correttamente.

Figura 5.24 – Riconoscimento del template dello spintore.

Passo 3. Selezione della guarnizione

Il sistema di visione è in grado anche di riconoscere i colori per discriminare la correttezza dell'esecuzione o per interagire con l'utente, utilizzando la funzione `RGBlob`.

In questo caso l'utente può scegliere di montare sul componente differenti guarnizioni a seconda dell'impiego della valvola. L'utente mostra al sistema di visione le guarnizioni, il cui colore in questo caso identifica un diverso materiale (e diverse proprietà meccaniche), e le sposta all'interno di un'area del campo visivo della webcam adibita a zona di interazione.

Il DSS interpreta e confronta le informazioni che gli sono trasmesse dal CVM e, secondo quanto descritto nel passo operativo, attiva l'ARM e gli impartisce istruzioni sulle informazioni da visualizzare all'utente. Il sistema è quindi in grado di interagire con l'utente e visualizzare maggiori dettagli sull'operazione corrente. Per concludere l'operazione, l'utente deve selezionare la guarnizione adatta e confermare la propria scelta, esponendo per un certo tempo il componente nell'area di controllo.
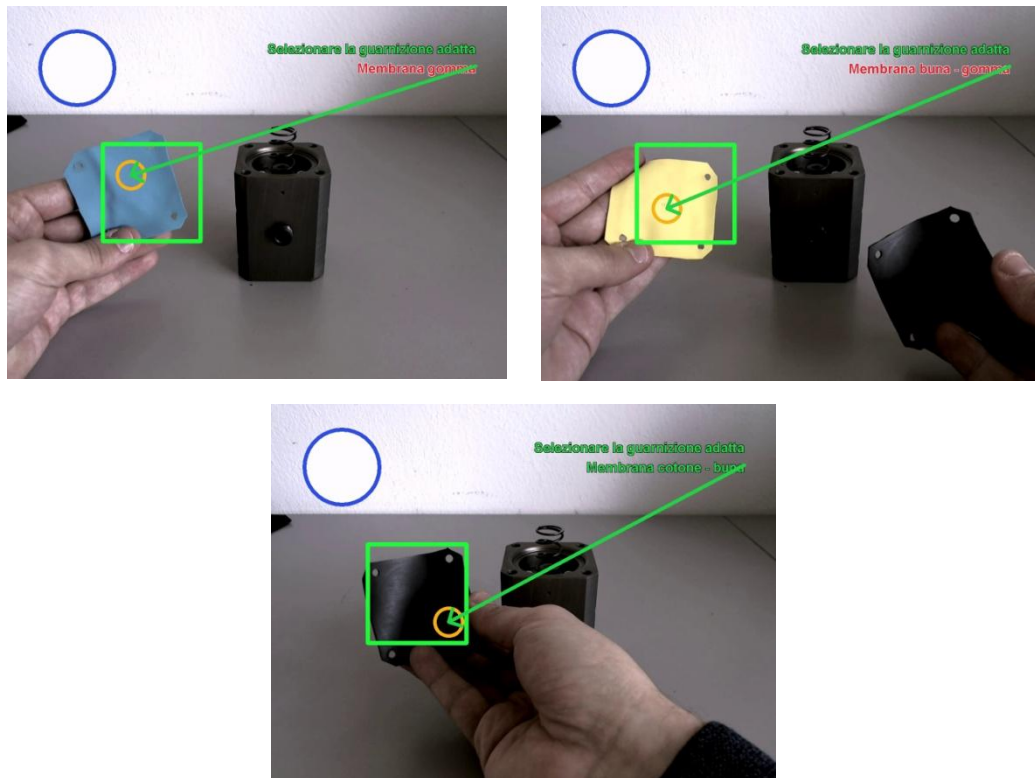
Figura 5.26 – Esecuzione del passo operativo corrente: il sistema identifica i colori delle membrane per effettuare la selezione.

Passo 4. Inserimento dello spintore

La guarnizione selezionata al passo operativo precedente viene montata dall'utente sullo spintore. Si è scelto di non monitorare questa operazione perché non coerente con le caratteristiche del sistema, che svolge la funzione di assistenza e formazione di un operatore e non è in grado, con la configurazione hardware adottata nel presente lavoro, di svolgere rilievi metrologici accurati, quali il serraggio di bulloni o la misura di tolleranze di accoppiamento, quali quelle necessarie per questa operazione.

La posizione in cui dovrà trovarsi lo spintore, affinché l'operazione si concluda correttamente, è fornita in coordinate relative al marker nella casella Check della procedura (**Fig. 5.19**, riga 8). A livello software, l'operazione che il sistema monitora è duale all'estrazione dello spintore. Il DSS comunica i parametri presenti nella casella di Input del passo operativo corrente al CVM per esplicitare i controlli sull'operazione da compiere ed i vincoli operativi. Anche l'interfaccia utente si presenta analoga alla precedente: l'ARM renderizza un cerchio che assume il colore rosso se il pezzo è all'esterno della propria sede, oppure verde se, invece, è stato correttamente inserito.
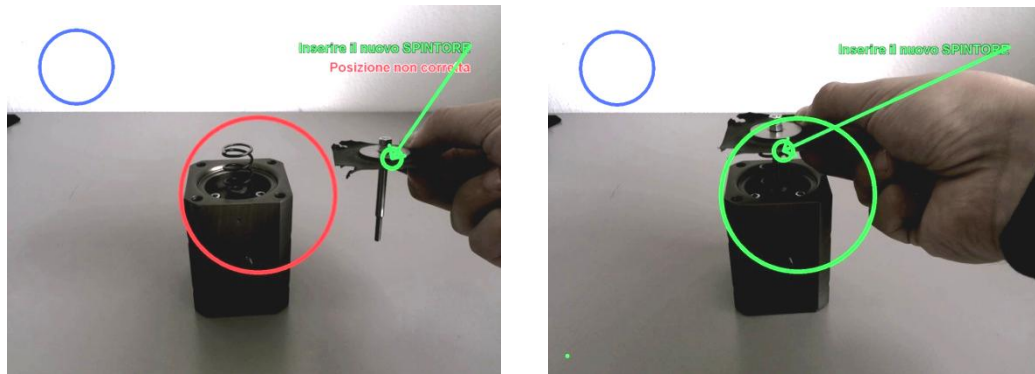
Figura 5.27 – Sequenza del passo operativo corrente: lo spintore deve essere inserito correttamente nella propria sede.

Figura 5.28 - Inseguimento del template.

Passo 5. Posizionamento del coperchio

Dopo aver inserito lo spintore la procedura prevede che sia montato nuovamente il coperchio della valvola. Il sistema di visione, applicando la funzione `PosE`, controlla che il marker selezionato nella procedura sia visibile e non in movimento. Il DSS attende dal CVM il permesso ad avanzare nella procedura.
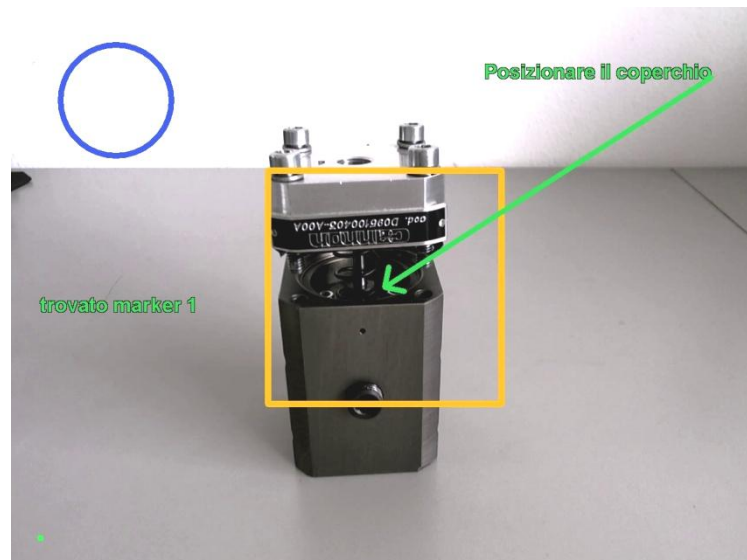
Figura 5.29 – Rendering dell'area di ricerca del coperchio.

Figura 5.30 – Il coperchio, correttamente posizionato, deve essere avvitato.

Passo 6. Assemblaggio del coperchio

Per concludere la procedura l'utente deve inserire ed avvitare le viti rimosse all'inizio della procedura. In analogia con le operazioni già compiute, il sistema cerca il marker naturale, identifica le coordinate della regione da controllare, effettua una differenza tra le immagini del flusso video per riconoscere variazioni dell'ambiente, applica un filtro binario ed estrae i blob dall'immagine. Utilizzando i parametri in input come filtro per selezionare le dimensioni dei blob da riconoscere ed evitare falsi positivi dovuti alle mani dell'operatore, il sistema riconosce che le viti siano state inserite ed avvitate.
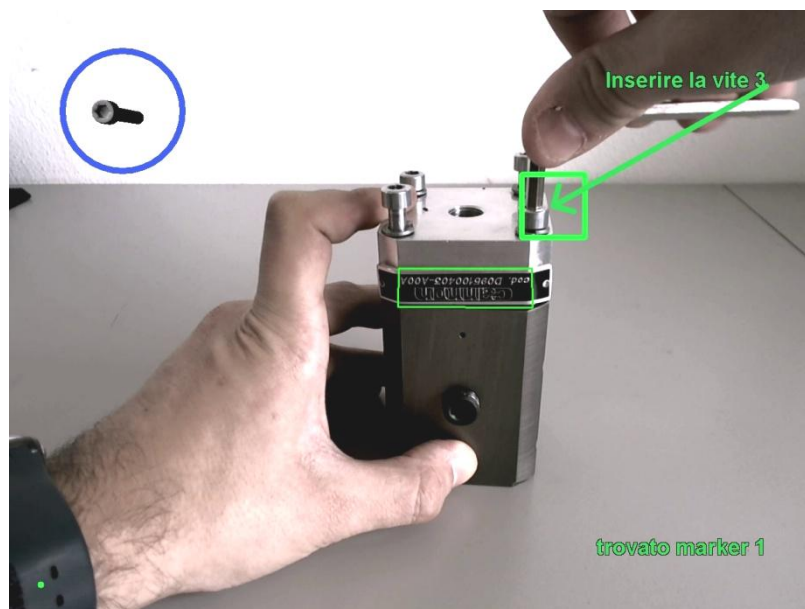
Figura 5.31 – Il marker naturale viene riconosciuto e si procede al montaggio del coperchio.

5.4. Caso 3 – Attrezzaggio di una macchina di misura tridimensionale

5.4.1. Fase istruttore: Creazione della procedura

Seguendo la stessa metodologia presentata nei casi applicativi precedenti (paragrafo 5.2), è stata redatta la procedura che il sistema dovrà eseguire e monitorare partendo da un'attenta analisi della sequenza dei compiti che l'operatore deve compiere. Incrementando ad ogni passo il livello di dettaglio dell'analisi, dalla sequenza di istruzioni elementari si arriva a definire un elenco di operazioni primitive di computer vision che il sistema deve gestire in base a quanto compilato nella procedura.
Il presente caso applicativo è stato condotto con l'obiettivo di attrezzare una macchina universale di misura equipaggiata con un tastatore laser per la misura delle tolleranze dimensionali di un componente cilindrico. Il modello del macchinario è una macchina MU 214-B prodotta dalla SIP Genevoise (**Fig.5.32**) e presente nel laboratorio di misura di ITIA-CNR.

Figura 5.32 – Macchina di misura universale MU 214-B Sip Genevoise.

L'utente deve posizionare in modo corretto le contropunte per il blocco del componente che si vuole misurare, facendole scorrere lungo una guida rettilinea. Poiché la procedura è specifica per la misura di un componente cilindrico l'utente deve quindi selezionare il pezzo fra alcune possibili alternative. Il sistema interviene inoltre se viene compiuto un errore per avvisare l'operatore e correggere, in tempo reale, lo svolgimento del passo operativo. Dopo il corretto posizionamento del pezzo, stretto fra le due contropunte, l'utente può accendere il laser per cominciare le fasi di misura.

La sequenza di operazioni identificata è schematizzata nella **Tabella 5.3**. Nella colonna *Operazione Manuale* è indicata la descrizione sintetica dell'operazione che l'utente deve compiere. Nella colonna *Descrizione dell'operazione* il singolo passo è analizzato con maggiore dettaglio e nella colonna *Definizione del Controllo Operativo del Sistema* è tradotto in una serie di operazioni primitive che il sistema deve eseguire per operare il controllo sulle azioni dell'utente. Nella colonna *Codice DSS* è indicato, infine, il codice sintetico riferito alla funzione del modulo di visione artificiale (capitolo 4) che si utilizza nel passo operativo.

Operazione manuale	Descrizione dell'operazione	Definizione del controllo operativo del sistema	Codice DSS
Inserimento della contropunta sinistra	Inserire la contropunta indicata e portarla in posizione	Identificare il pezzo corretto e verificare le sue coordinate nell'immagine	TMatch
Inserimento della contropunta destra	Inserire la contropunta indicata e portarla in posizione	Identificare il pezzo corretto e verificare le sue coordinate nell'immagine	TMatch
Selezione del pezzo	Afferrare il pezzo corretto	Identificare componenti diversi e verificare la selezione dell'utente	TMatch
Posizionamento del pezzo fra le due contropunte	Inserire il pezzo fra le due contropunte	Rilevare la presenza del pezzo nella posizione indicata	BlobPG
Accensione del laser	Ruotare la chiave di accensione	Attendere che il led si accenda	RGBlob

Tabella 5.3 – Codifica delle operazioni: dal manuale di istruzioni alla realtà aumentata.

Il diagramma di flusso che rappresenta le regole di funzionamento seguite nella progettazione della procedura è rappresentato in **Fig. 5.33**. Ogni rilevamento dello stato del sistema e dei marker naturali è effettuato dal sistema di visione e controllato dal modulo decisionale (DSS) che indica come procedere.
Il DSS, infatti, legge le istruzioni contenute nella procedura ed interroga il modulo di visione (CVM) per controllare lo stato di avanzamento delle operazioni dell'utente. Sulla base della risposta del CVM restituisce un feedback visuale all'utente sfruttando le funzionalità del modulo di realtà aumentata (ARM).
L'ultimo passo della fase istruttore è la redazione della procedura. Attraverso lo studio preliminare del problema, di cui si è appena discusso, e l'utilizzo dell'interfaccia utente dell'applicazione istruttore.exe, si giunge al salvataggio nel database delle regole che il sistema dovrà seguire nello svolgimento della fase operatore (**Fig. 5.34**).
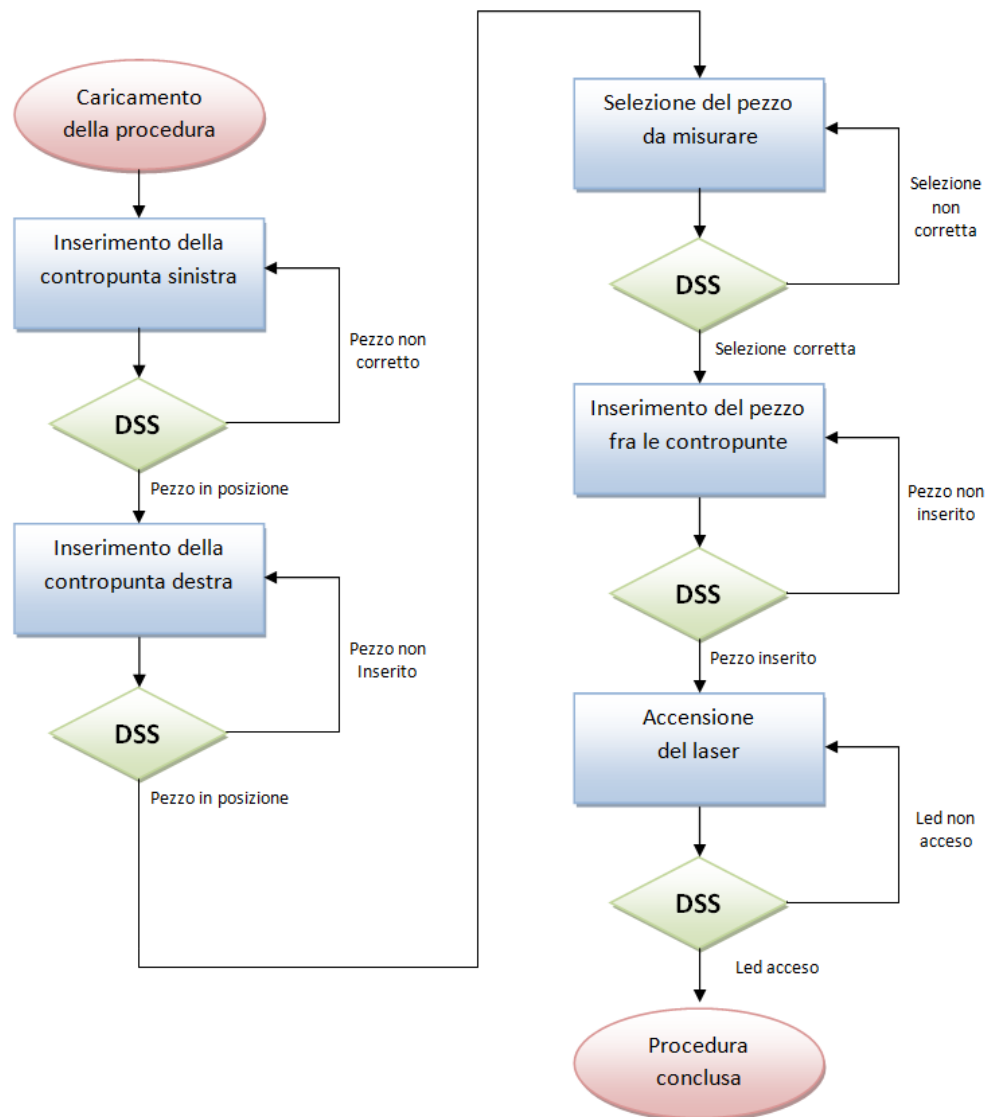
Figura 5.33 – Diagramma di flusso della procedura del caso applicativo: il modulo di supporto decisionale (DSS) legge le istruzioni contenute nella procedura ed interroga il modulo di visione (CVM) per controllare lo stato di avanzamento delle operazioni dell'utente.

5.4.2. Fase operatore: Esecuzione della procedura

Il sistema in modo autonomo esegue la procedura (**Fig. 5.34**) ed interpreta le modifiche apportate dall'utente allo stato del sistema per prendere decisioni e modificare di conseguenza il flusso logico del processo schematizzato nel paragrafo precedente.

	Tipo	Descrizione	Opera	Ref Marker	Input	Check	Modelli3D	Tra			I			path	P	Verifica	Condiz
1	☐	Inserire la contropunta sinistra	Tmatch	./SIP2/SIP2_ma	./SIP2/1/SIP2_	-400 10	punta1	-0.3	0	0	0	0	0	./modelli3D/pun	1	☑	☐
2	☑	Contropunta non corretta	Tmatch	-1	-1	1	punta2	-0.3	0	0	0	0	0	./modelli3D/pun	1	☐	☐
3	☑	Spostare la contropunta nella cor	Tmatch	-1	-1	2	punta1	-0.3	0	0	0	0	0	./modelli3D/pun	1	☐	☑
4	☐	Inserire la contropunta destra	Tmatch	-1	./SIP2/4/SIP2_	1 -170 10	punta2	-0.15	0	0	0	0	0	./modelli3D/pun	1	☐	☐
5	☐	Prendere il pezzo da misurare	Tmatch	0	./SIP4/2/SIP4_	0 150 350 140 1										☐	☐
6	☐	Tornare alla macchina	PosE	./SIP4/SIP4_ma	640 480 600	0.05										☐	☐
7	☐	Inserire il pezzo da misurare	BlobPG	-1	0.3 1	-250 10 200	v_box	-0.9	0.6	-2	0	0	0	./modelli3D/box	0	☐	☐
8	☐	Accendere il laser	RGBlob	./SIP3/SIP3_ma	0.5 green	130 60 40										☐	☐
9	☐	Regolare lo strumento ed esegui	WaitOP	./SIP3/SIP3_ma	15	15										☐	☐

Figura 5.34 – Procedura del caso applicativo 3.

In questo caso applicativo si è scelto di inserire nel flusso video i modelli 3D delle contropunte, realizzati con il software 3D Studio Max (**Fig. 5.35**, **Fig. 5.36**). I modelli permettono una comprensione più immediata delle informazioni che, in questo modo, possono essere visualizzate senza che l'utente distolga lo sguardo dal centro di attenzione principale, cioè l'area in cui deve eseguire il proprio compito.
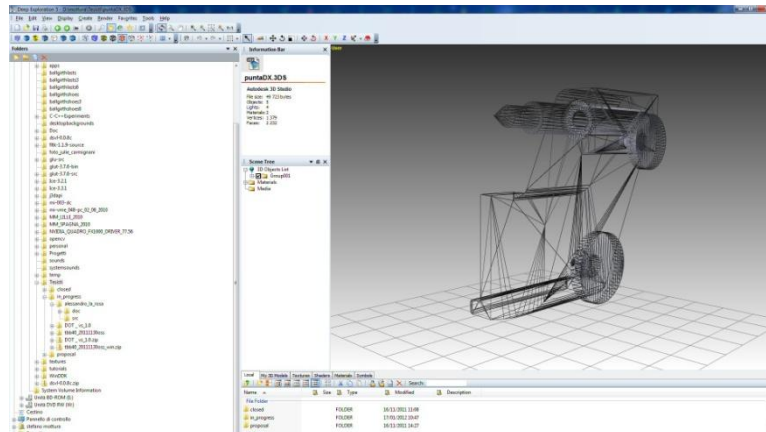
Figura 5.35 – Visualizzazione wireframe del modello 3D.
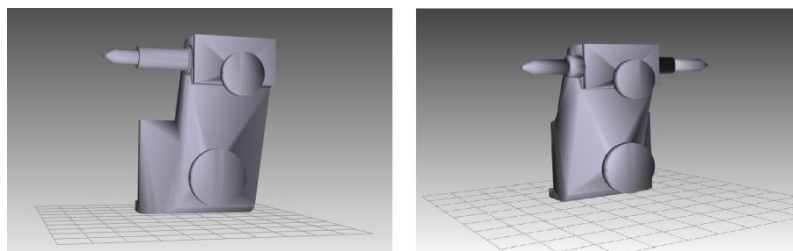
Figura 5.36 – Modelli 3D realizzati.

Passo 1. Inserimento della contropunta sinistra

Il sistema individua il marker naturale selezionato e, a partire dalla sua posizione nello spazio, costruisce un sistema di riferimento a cui associare tutti gli elementi presenti nella scena osservata dalla telecamera. A causa delle dimensioni dell'area di lavoro in rapporto agli oggetti da movimentare e della configurazione di telecamera mobile montata sulla testa dell'utente (possibili movimenti repentini della telecamera) si è scelto di utilizzare come marker l'immagine ingrandita di un particolare della macchina (**Fig. 5.37**). Il sistema di tracciamento, descritto nel capitolo 4 e sviluppato nel CVM, cerca il marker all'inizio dell'esecuzione del passo operativo e ne restituisce la posizione nello spazio (6 DOF) dopo un numero di acquisizioni tali da garantire la stabilità dei valori calcolati (paragrafo 4.2.1). Il DSS legge dalla procedura i percorsi dei modelli 3D da visualizzare e la loro posizione rispetto al marker, invia poi questi dati all'ARM per visualizzare sul display, di volta in volta, le informazioni necessarie alla corretta esecuzione del passo operativo.
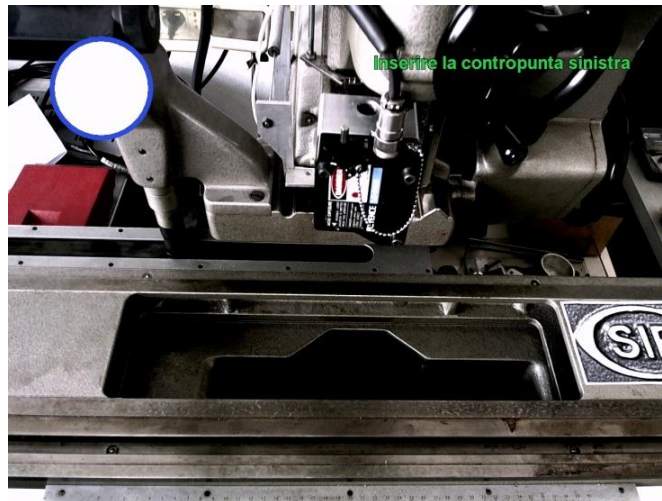
Figura 5.37 – Setup del passo operativo con marker a bordo macchina.

Per identificare la contropunta corretta viene utilizzata la funzione `TMatch()` che cerca nell'immagine un template registrato durante la fase istruttore.
A questa operazione è associata una verifica aggiuntiva, poiché è gestito il caso in cui l'utente inserisca una contropunta errata.
Il DSS legge le possibili configurazioni dal file della procedura, le confronta con quelle restituite dal CVM e comunica cosa visualizzare all'ARM, per guidare l'utente nel passo operativo o per correggerlo in caso di errore.
Sul display vengono renderizzate le istruzioni testuali, il modello 3D della contropunta da inserire ed un rettangolo (collegato al testo con una freccia) che individua la sua corretta posizione nell'immagine.
Nel caso in cui l'utente inserisca la contropunta errata, il modello virtuale viene colorato di rosso (**Fig. 5.38**). Se, invece, viene inserita la contropunta sinistra ma non viene spostata nella posizione indicata dal rettangolo rosso, il modello 3D assume il

colore verde (**Fig. 5.39**). Infine, se la contropunta viene inserita e spostata all'interno del rettangolo, questi si colora di verde e non si ritiene necessario visualizzare il modello 3D (**Fig. 5.40**).
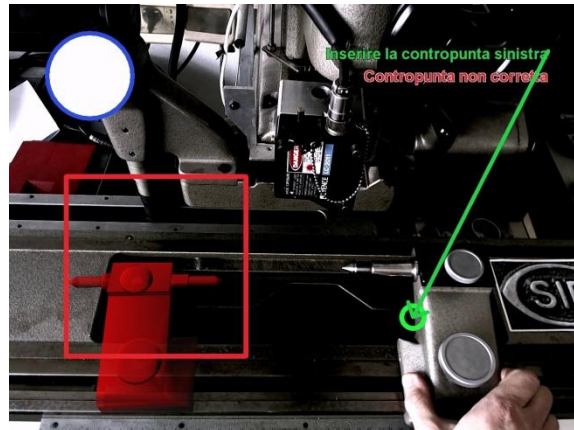
Figura 5.40 – Inserimento della contropunta errata.
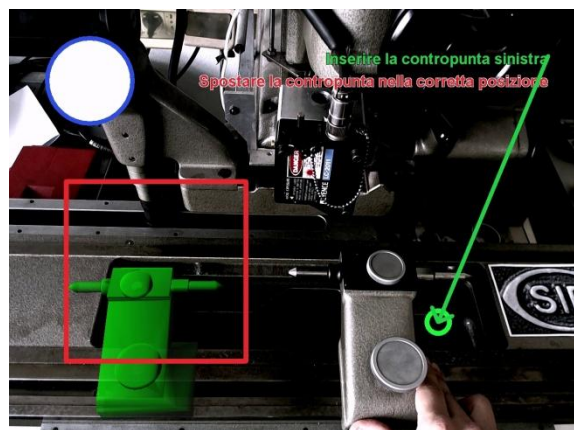
Figura 5.38 – Inserimento della contropunta corretta .
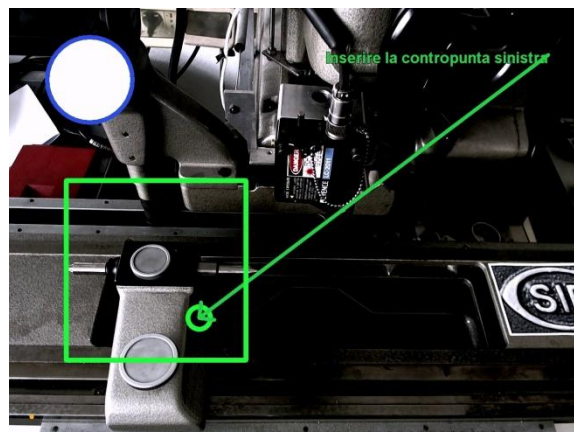
Figura 5.39 – Corretto posizionamento della contropunta.

Passo 2. Inserimento della contropunta destra

Come per l'operazione precedente, il DSS coordina lo scambio di dati ed informazioni tra i moduli di Visione (direttamente collegato alla telecamera) e Realtà Aumentata (con accesso diretto al display).

L'utente viene assistito nell'operazione con descrizioni testuali delle istruzioni, frecce, modelli 3D e rettangoli contestuali colorati. Ad un'operazione corretta è associato il colore verde, ad un'operazione errata il colore rosso.

In **Fig. 5.41**, **Fig. 5.42**, **Fig. 5.43** si osservano, analogamente al passo operativo precedente, le possibili configurazioni dell'ambiente di lavoro che il DSS può interpretare.
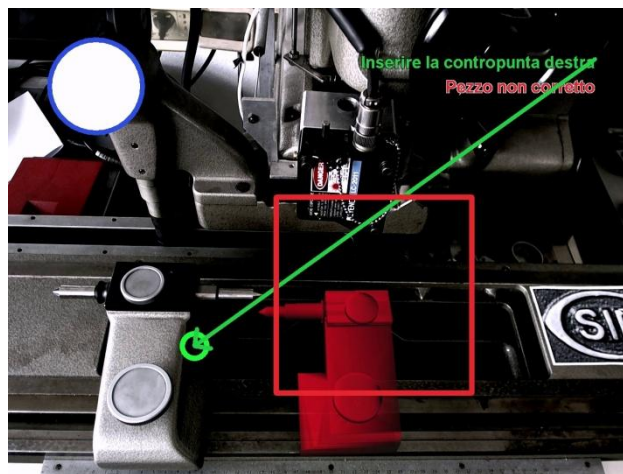
Figura 5.41 – Mancato inserimento del pezzo
(modello 3D rosso, rettangolo rosso, testo dell'istruzione).
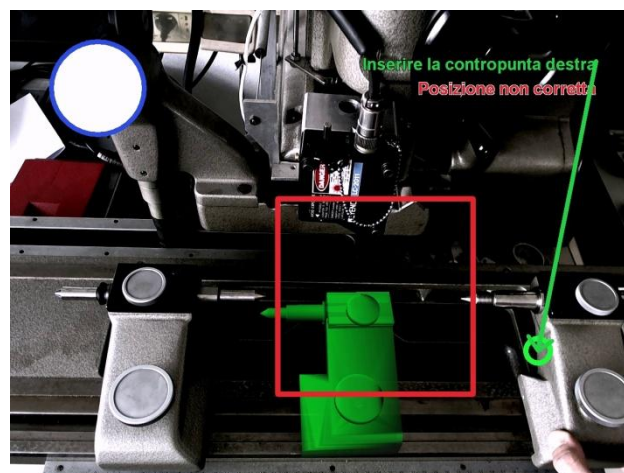
Figura 5.42 – Inserimento del pezzo in una posizione non corretta (modello 3D verde, rettangolo rosso, testo rosso che segnala l'operazione incompleta).
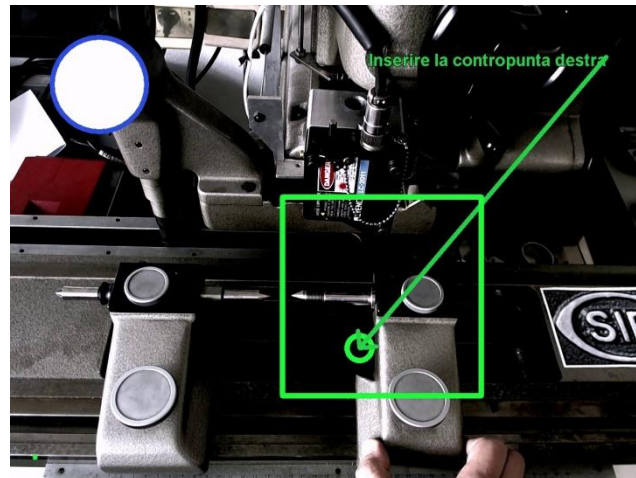
Figura 5.43 – Inserimento del pezzo nella posizione corretta (rettangolo verde, assenza di segnali di errore).

Passo 3. Selezione del pezzo da misurare

La procedura in esecuzione è specifica per un pezzo dalla forma cilindrica.
L'utente deve selezionare il pezzo corretto fra quelli presenti sul tavolo da lavoro e confermarne la selezione alla telecamera muovendo il pezzo all'interno di un'area dell'immagine adibita alla scelta utente e rappresentata da un cerchio rosso in **Fig. 5.44**.
Il CVM identifica gli oggetti presenti nella scena e permette all'utente la selezione del solo pezzo congruo. L'ARM restituisce, in tal senso, un feedback visuale, colorando di verde cerchio dell'interfaccia (**Fig. 5.45**) solo quando l'utente vi sposta all'interno il pezzo corretto.
Alla fine di questa operazione l'utente è invitato a rivolgere nuovamente il proprio sguardo alla macchina. Il DSS considera correttamente completato il passo operativo solo quando il CVM gli indica l'effettivo riconoscimento del marker naturale iniziale (**Fig. 5.46**).

Passo 4. Inserimento del pezzo fra le due contropunte

Il pezzo selezionato al passo operativo precedente deve essere montato dall'utente fra le due contropunte, prima di poter eseguire la misura.
Il rilevamento del pezzo è effettuato dal CVM con la funzione `BlobPG` (capitolo 4). Sul display l'ARM comunica all'utente la posizione in cui inserire il pezzo attraverso il rendering di un rettangolo verde e di una freccia che collega le istruzioni testuali al centro del rettangolo, situato tra le due contropunte (**Fig. 5.47**).
Il DSS comunica i parametri presenti nella casella di Input del passo operativo corrente al CVM per esplicitare i controlli sull'operazione da compiere ed i vincoli operativi.
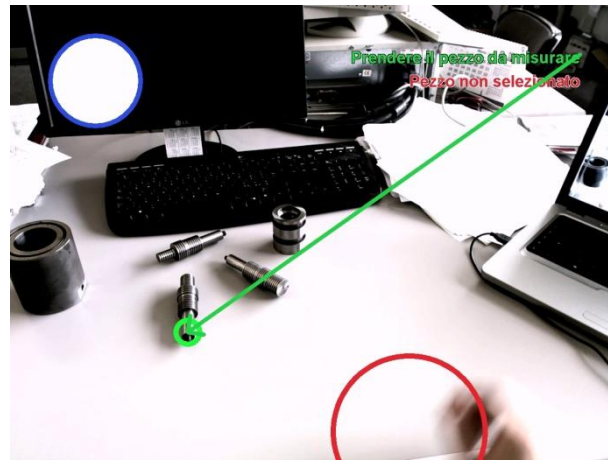
Figura 5.45 – Rilevamento del pezzo. L'area di selezione è rappresentata dal cerchio rosso in basso a destra.
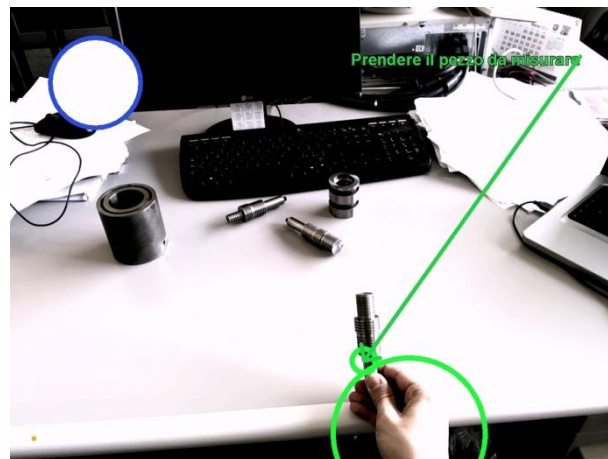
Figura 5.46 – Selezione del pezzo da misurare.
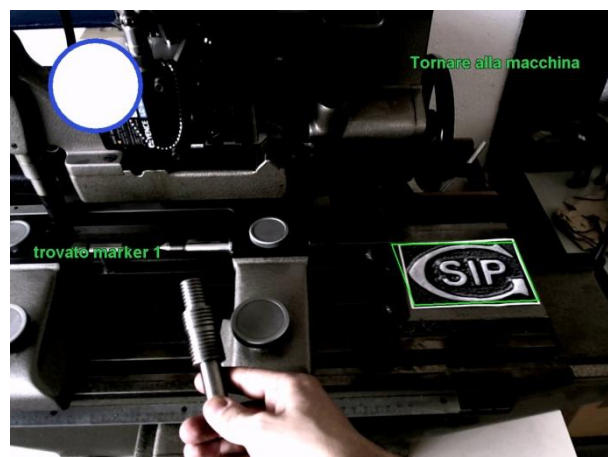
Figura 5.44 – Ricerca del marker naturale per confermare il ritorno dell'utente alla postazione di misura.
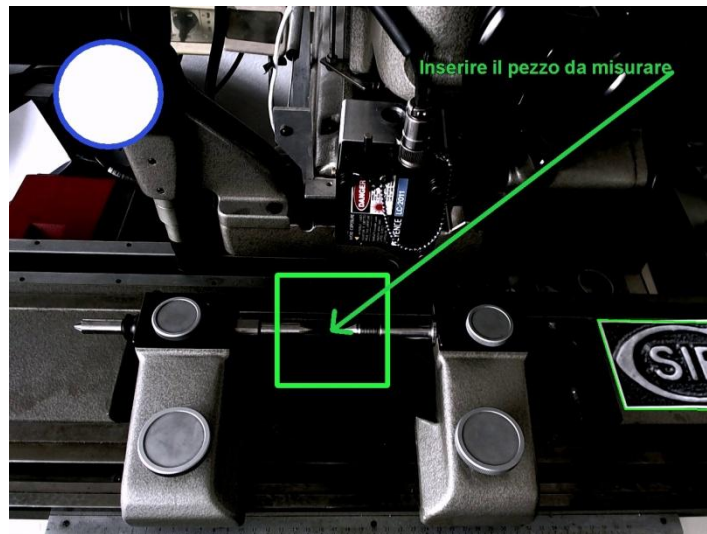
Figura 5.47 – Localizzazione della sede in cui si deve inserire il pezzo.
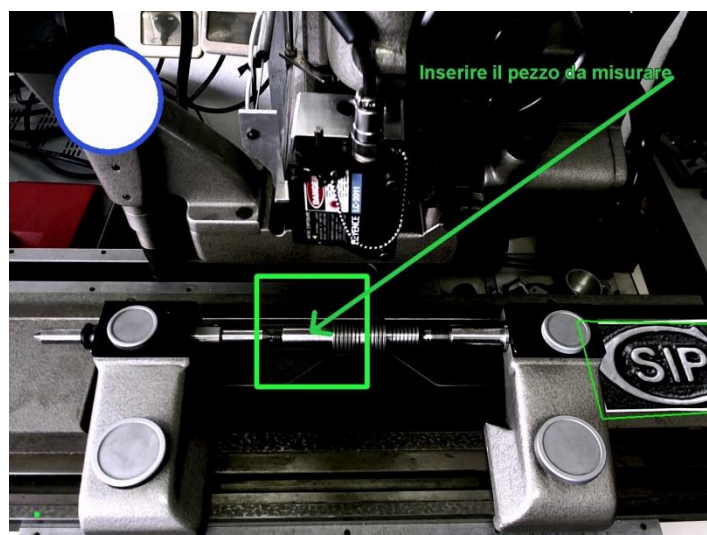
Figura 5.48 – Rilevamento del corretto inserimento del pezzo.

Passo 5. Accensione del laser

Dopo aver inserito il pezzo la procedura prevede che sia acceso il laser.
L'utente si muove verso un'altra postazione. Al CVM viene fornito dal DSS un nuovo marker da rilevare (**Fig. 5.49**) che corrisponde all'interfaccia del dispositivo di alimentazione del laser.
Nelle caselle di *Input* e *Check* della procedura sono stati definiti i parametri necessari al completamento dell'operazione, corrispondenti alle posizioni (dall'origine del marker) della chiave di accensione e del led colorato da controllare con la funzione

`RGBlob`. Intorno alla chiave, che l'utente deve ruotare per accendere il laser, l'ARM visualizza un cerchio verde. Una freccia collega il centro del cerchio al testo dell'istruzione che l'utente deve compiere. In parallelo, il CVM monitora il led rosso, racchiuso in un rettangolo verde sul display, in attesa della sua accensione.
Il DSS attende la conferma dal modulo di visione per completare l'esecuzione della procedura (**Fig. 5.50**).

Figura 5.49 – Marker naturale utilizzato per il passo operativo corrente.
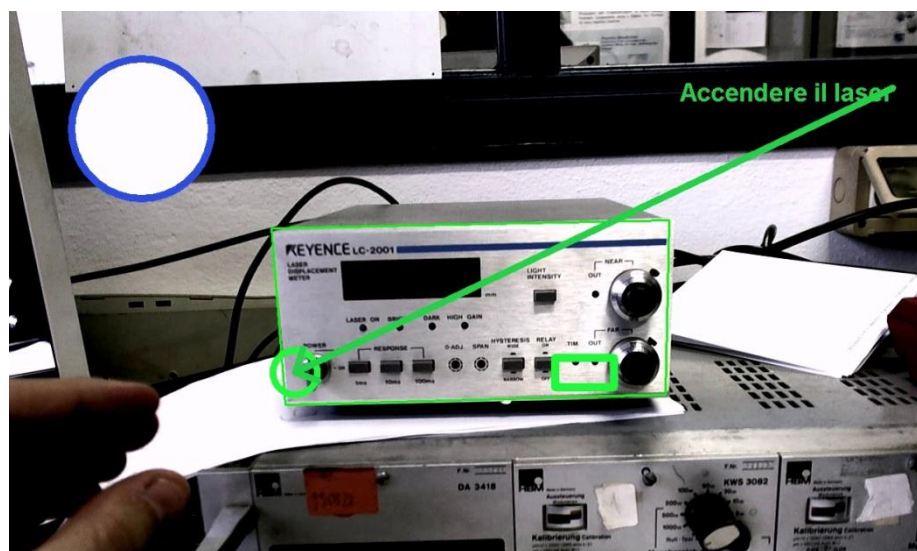
Figura 5.50 – Riconoscimento del marker naturale e localizzazione delle aree di interesse. Nel cerchio (a sinistra) è presente l'interruttore per accendere il laser. Il rettangolo (a destra) indica al sistema dove verificare la presenza di un led colorato acceso.

Conclusioni e sviluppi futuri

Il lavoro di Tesi ha condotto allo sviluppo di un sistema che, in modo completamente autonomo, fornisce all'utente indicazioni sulle operazioni da compiere ed informazioni sul loro stato di completamento. Il sistema è in grado di adattare le proprie valutazioni alle modifiche apportate dall'utente all'ambiente di lavoro, grazie all'implementazione di funzionalità di computer vision per il riconoscimento e l'inseguimento di features naturali. La comunicazione con l'utente avviene attraverso tecniche di Realtà Aumentata che migliorano la qualità del contributo informativo anche tramite l'impiego di modelli tridimensionali sovrapposti in tempo reale al flusso video osservato da una webcam.
Il funzionamento del sistema è controllato da una sequenza ordinata e parametrica di operazioni che viene programmata con un'apposita applicazione. L'intuitività della logica di programmazione e la flessibilità delle funzionalità sviluppate hanno consentito di utilizzare il sistema in contesti differenti. Sono stati, infatti, sviluppati tre casi applicativi: la riparazione di una membrana di tenuta in un componente meccanico, la sostituzione di una scheda elettronica in un personal computer, l'attrezzaggio di una macchina di misura tridimensionale.

L'esecuzione dei casi applicativi ha mostrato come l'utente sul campo sia stato libero di eseguire le istruzioni visualizzate senza particolari vincoli su gesti e movimenti da compiere all'interno dell'area di lavoro, come si è potuto osservare nelle riprese in soggettiva della webcam.
I casi applicativi hanno inoltre evidenziato la prontezza della risposta del sistema alle variazioni apportate dall'utente all'ambiente di lavoro durante il suo operato. Le caratteristiche di affidabilità del sistema si fondano sull'interpretazione degli input e sulla modifica conseguente del flusso logico del processo da monitorare; il sistema riesce dunque ad adattarsi alle decisioni dell'operatore che si avvale del suo supporto. È perciò possibile, a partire da queste prove applicative, affermare che, con questa tesi, è stato fornito ai tecnici addetti alla manodopera un sistema in grado di assisterli con versatilità, adattabilità e precisione.

In futuro la validazione del sistema in reali scenari industriali consentirebbe di valutarne il funzionamento "sul campo" unitamente alla raccolta delle impressioni degli utenti a cui il sistema effettivamente si rivolge.

Un'interessante linea di sviluppo è quella volta al riconoscimento dei gesti dell'utente o al tracciamento delle mani dell'operatore per migliorare ulteriormente la qualità

dell'interazione fra operatore e sistema, utilizzando, ad esempio, i gesti dell'utente per istruire direttamente il sistema e per comunicare scelte e impartire comandi che potrebbero modificare il flusso logico della procedura.
Infine una prospettiva interessante è quella di fare il porting del software, ora realizzato su personal computer, su dispositivi mobili quali, ad esempio, un *tablet Pc* che, grazie alla sua elevata trasportabilità, potrebbe sfruttare la telecamera integrata per una più agevole programmazione delle regole della procedura: ad esempio, permettendo di inquadrare facilmente i marker naturali presenti nell'ambiente. Il *touchscreen*, inoltre, potrebbe incrementare la semplicità di controllo grazie ad un interfaccia utente più intuitiva e priva di periferiche e cavi.

Appendice A.
La formazione dell'immagine

Il dispositivo più semplice per la formazione di un'immagine di una scena tridimensionale è la fotocamera *stenopeica* o *pinhole camera* (**Fig. A.1**), un sistema visivo artificiale, in cui la luce passa attraverso un piccolissimo foro (sistema ottico) e forma su un piano sensibile (sensore) un'immagine invertita della scena.
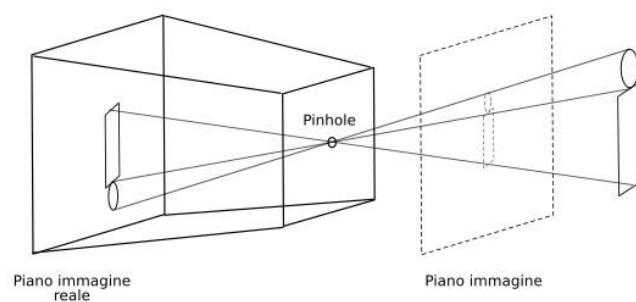
Figura A.1 – Formazione dell'immagine nel modello pinhole.

Da un punto di vista matematico l'immagine può essere ricostruita tracciando dei raggi rettilinei che partono da punti diversi della scena ed intersecano il piano immagine passando attraverso il foro.
E' quindi possibile costruire un modello geometrico per rappresentare la formazione dell'immagine: la proiezione prospettica.
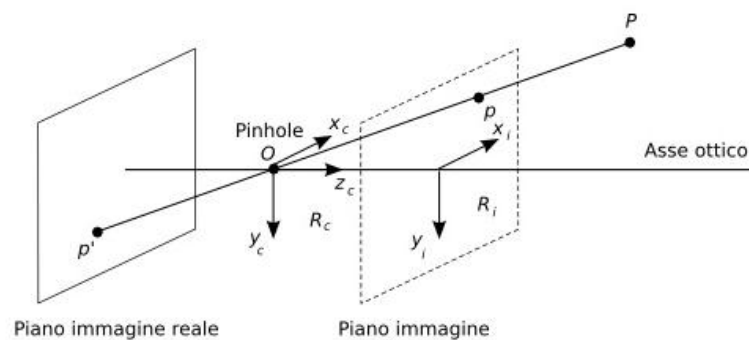
Figura A.2 – Modello geometrico della proiezione prospettica.

Per ottenere immagini nitide è necessario ridurre la dimensione del *pinhole*, ma in questo modo l'immagine sarebbe poco luminosa. Al contrario, per aumentare la luminosità sarebbe necessario ingrandire il foro stenopeico, tuttavia a ogni punto dell'immagine non corrisponderebbe più un solo raggio luminoso ma un cono di raggi luminosi convergenti che darebbero come risultato immagini offuscate.

Il compromesso tra le due situazioni descritte si ottiene utilizzando un obiettivo, cioè un sistema di lenti convesse o convesse e concave. Le lenti sono caratterizzate da:

- campo di visione: porzione di spazio realmente proiettato sulla retina della telecamera;
- luminosità, ovvero quantità di luce raccolta;
- profondità di fuoco: distanza entro la quale gli oggetti sono sufficientemente a fuoco; dipende dal rapporto tra la lunghezza focale e il diametro delle lenti.

Il più semplice sistema ottico che raccoglie i principi di base di un obiettivo è quello delle lenti sottili. Esse sono caratterizzate da un asse ottico, passante per il centro della lente *O* e perpendicolare al piano della lente, e due fuochi F_i e F_r, cioè due punti dell'asse ottico a distanza *f* (lunghezza focale) da *O* che si trovano nei due lati opposti della lente.
Il funzionamento delle lenti sottili (**Fig. A.3**) è riassunto da due affermazioni:

- ogni raggio, parallelo all'asse ottico, entrante nella lente esce dall'altro lato della lente passando per il fuoco
- ogni raggio, passante per il fuoco, entrante nella lente esce dall'altro lato della lente parallelo all'asse ottico.
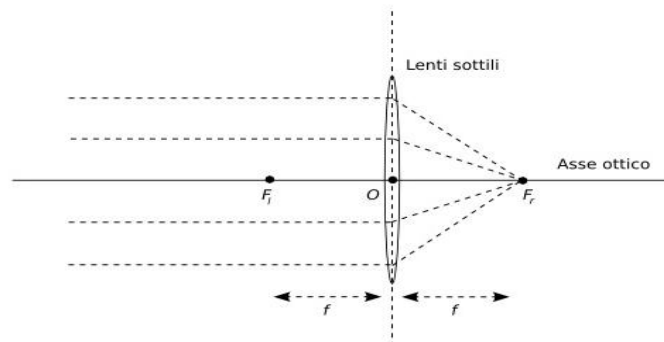
Figura A.3 – Lenti sottili.

L'utilizzo del modello a lenti sottili modifica leggermente l'equazione che lega un punto 3D nello spazio al suo corrispondente punto 2D nell'immagine. Questo effetto non rientrerà però nel modello basato sulla proiezione prospettica utilizzato per la calibrazione della telecamera. In realtà le lenti sottili sono un'approssimazione di un sistema ottico reale molto più complesso che consiste in un numero di lenti con diverso spessore e differente curvatura.
Verrà, tuttavia, considerata la presenza delle lenti come fonte di distorsione dell'immagine.
Il modello *pinhole* è ritenuto un valido modello descrittivo della telecamera perché, benché sia semplice e ideale, fornisce un'approssimazione accettabile del processo di formazione dell'immagine con convenienza anche dal punto di vista matematico e computazionale.

Appendice B.
Strumenti per lo sviluppo del sistema: le librerie

La necessità di realizzare un'applicazione che esegua in tempo reale complessi algoritmi di elaborazione dell'immagine ha spinto verso la ricerca di librerie di funzioni che fornissero un'implementazione adeguata di molti algoritmi di elaborazione delle immagini e di visione artificiale. Tra le numerose proposte presenti in letteratura, di cui ampiamente discusso nei capitoli precedenti, si sono scelte le seguenti librerie durante la fase di progettazione del software del sistema: OpenCV, DOT e GIOVE.
La scelta è stata motivata dalle caratteristiche open source e multipiattaforma e dalla folta comunità di utilizzatori di alcune delle librerie presentate (OpenCV e DOT), nonché dalla possibilità di implementare robusti algoritmi real-time.
Gli algoritmi di elaborazione delle immagini, acquisizione video da webcam e tracking sono implementati nella libreria OpenCV. Per il rendering real-time dei modelli 3D è stata utilizzata la libreria GIOVE, un framework di librerie sviluppato all'interno di ITIA (CNR) per applicazioni di grafica 3D realtime.
La libreria DOT è stata utilizzata per sviluppare una parte del tracking e l'algoritmo di template matching.

1. La libreria Intel® OpenCV

Nata nel 1999 da Intel e giunta allo stato di sviluppo 2.4.2 (ottobre 2012), OpenCV raccoglie diverse centinaia di funzioni C/C++ di computer vision che forniscono un'interfaccia di programmazione di medio-alto livello per tutti gli strumenti analitici di ricerca nel campo della visione artificiale.
La libreria OpenCV è stata sviluppata per essere computazionalmente efficiente con particolare attenzione alle applicazioni real-time: è stata ottimizzata per sfruttare le potenzialità dei processori multi-core e per gestire numerosi formati di rappresentazione dei dati secondo standard aperti e condivisi, al fine della gestione di immagini e della manipolazione di matrici.
Uno dei punti di forza di questa libreria è il fatto di essere multipiattaforma ed *open source* e di essere una tra quelle particolarmente diffuse nella comunità scientifica. Questo garantisce piena compatibilità sia su piattaforme Windows che Unix ed un costante miglioramento nell'implementazione delle funzioni: potenzialmente chiunque può dare un contributo segnalando/correggendo bug ed implementando

nuovi algoritmi, che rispettino i vincoli delle licenze BSD imposte dagli sviluppatori del progetto[5].
Durante lo svolgimento di questa tesi, inoltre, è stato segnalato e risolto un bug all'interno di una funzione della libreria, proponendo una soluzione che ha permesso di risparmiare circa 30 ms durante l'esecuzione on-line del sistema proposto.
La libertà di sviluppo concessa dagli sviluppatori ha consentito la nascita di una grande comunità di utilizzatori che include sia prestigiosi centri di ricerca che grandi compagnie commerciali, quali, ad esempio, Stanford, MIT, CMU, Cambridge, INRIA, CNR, IBM, Microsoft, Intel, SONY, Siemens, Google (Google's Street View), e molti altri.
Grazie alle molteplici funzionalità, OpenCV è stata utilizzata in numerose applicazioni, prodotti e ricerche, fin dalla sua prima versione. Queste applicazioni includono l'elaborazione digitale di immagini (riduzione del rumore, operazioni morfologiche, image stitching), riconoscimento e analisi di oggetti, sistemi di videosorveglianza, sistemi di ispezione industriale, calibrazione delle telecamere, guida robot e visual servoing.
Infine, bisogna ricordare che OpenCV è nata dalla sezione di ricerca di Intel. Questo significa che le applicazioni che fanno uso dell'interfaccia di programmazione da essa fornita sono ottimizzate per processori Pentium, in quanto possono sfruttare le *Integrated Performance Primitives* (IPP), librerie prodotte da Intel allo scopo di ottimizzare le applicazioni multimediali.

Panoramica sulle componenti di OpenCV

Una volta installato il pacchetto fornito dagli sviluppatori di OpenCV e predisposto l'ambiente di lavoro, si può iniziare a lavorare con le funzioni offerte: a tale scopo viene di seguito svolta una panoramica sulla struttura della libreria.
L'indirizzo internet di riferimento, al quale è possibile trovare l'intera documentazione (aggiornata sempre all'ultima versione) è http://opencv.itseez.com.
OpenCV ha una struttura modulare che include numerose librerie statiche o condivise.
I principali moduli sono:

- **Core** – modulo in cui vengono definiti tipi, strutture di allocazione dei dati e funzioni di base, quali ad esempio le strutture `Mat` (per immagini e matrici), funzioni per la manipolazione di vettori o che regolano la gestione della memoria.
- **Imgproc** – modulo per l'elaborazione dell'immagine, include funzioni lineari e non lineari per trasformazioni geometriche e morfologiche o conversioni da diversi spazi di colore.
- **Video** – modulo per l'analisi di video. Include gli algoritmi di calcolo per il tracking di oggetti in movimento.

[5] le licenze BSD riflettono l'idea più ampia possibile del dono liberale: chiunque può fare ciò che meglio crede del programma rilasciato ed acquisito con l'unico dovere di citare l'autore. Non si può ridistribuire il codice con un'altra licenza e assieme alla ridistribuzione dei file binari deve essere allegata la licenza.

- **Calib3d** – Modulo per la calibrazione di sistemi mono- e multi-camera, racchiude gli algoritmi per il calcolo della posizione e della proiezione prospettica di oggetti e di ricostruzione tridimensionale.
- **Features2d** – in questo modulo sono definite le funzioni per l'estrazione di features e la ricerca delle corrispondenze per il tracking di oggetti 2D.
- **Objdetect** – modulo in cui sono definite le funzioni per il riconoscimento e l'analisi di oggetti di classi predefinite (ad esempio, le funzioni di riconoscimento dei volti, degli occhi, ecc.)
- **Highgui** – modulo che gestisce l'interfaccia grafica di alto livello, permette di salvare e caricare immagini, acquisire flusso video da una telecamera, oppure di creare finestre per la visualizzazione su schermo delle immagini.

2. La libreria DOT: Dominant Orientation Templates

Dal 2007 il dipartimento CAMPAR (Computer Aided Medical Procedures & Augmented Reality) dell'Università di Monaco ha iniziato lo sviluppo di software finalizzato alla ricerca di oggetti 3D all'interno di una scena ed alla stima della camera pose in real-time.

La libreria denominata Dominant Orientation Templates [6], sviluppata da S. Hinterstoisser (et al.) e rilasciata nel giugno 2010 sia sotto piattaforma Windows che Linux, implementa un metodo per la rilevazione real-time di oggetti 3D non texturizzati.

Tra le funzionalità che questa libreria offre, c'è la possibilità di scegliere quali oggetti cercare e di poter istruire il software secondo due modalità di interazione differenti.

Si può effettuare sia una registrazione dei vari oggetti, chiamati template, in un momento diverso all'esecuzione dell'applicazione in real-time e caricare, all'occorrenza, i template salvati, sia eseguire registrazione e tracking dei template eseguendo in tempo reale il programma.

La libreria DOT sfrutta alcune funzioni di OpenCV e richiede l'utilizzo delle IPP intel, per prestazioni più efficienti. Alcune funzioni di questa libreria sono state inserite nella versione 2.4 di OpenCV.

3. GIOVE (Graphics & Interaction for OpenGL-based Virtual Environment)

All'interno di ITIA (Istituto di Tecnologie Industriali ed Automazione) del Consiglio Nazionale delle Ricerche (CNR) è stata sviluppata la libreria GIOVE (Graphics and Interaction for OpenGL-based Virtual Environments): un insieme di strumenti software, scritti in linguaggio C++, per lo sviluppo di applicazioni di Realtà Virtuale basate sulle librerie standard OpenGL.

Con OpenGL si definisce una API (Application Programming Interface) per più linguaggi e piattaforme per scrivere applicazioni che producono computer grafica realtime 2D e 3D.

[6] http://campar.in.tum.de/personal/hinterst/index/index.html

Appendice C.
Validazione della Calibrazione della telecamera

Dopo aver implementato la procedura di calibrazione della telecamera descritta con le librerie OpenCV, si è scelto di effettuare un confronto dei risultati, per verificarne la correttezza, provando ad eseguire la calibrazione della telecamera con il Calibration Toolbox di Matlab, che implementa il metodo di calibrazione di Zhang.
Il toolbox[7] è stato sviluppato dall'università finlandese di Oulu e richiede in ingresso una serie di fotogrammi contenenti una scacchiera ripresa da diverse angolazioni. A differenza del programma che utilizza le librerie OpenCV, il toolbox di Matlab richiede la selezione manuale di quattro corner per ogni scacchiera. Questa operazione richiede tempo e molta attenzione da parte dell'utente che, in caso di errore, dovrà eseguire nuovamente l'intera procedura di calibrazione. La criticità maggiore si ha nel dover rispettare lo stesso ordine di selezione dei corner per ogni scacchiera (ad esempio in senso antiorario partendo dal corner in alto a sinistra) per poter ottenere risultati di calibrazione affidabili.
Utilizzando le immagini salvate dal programma precedentemente descritto, si è lanciata la procedura di calibrazione.
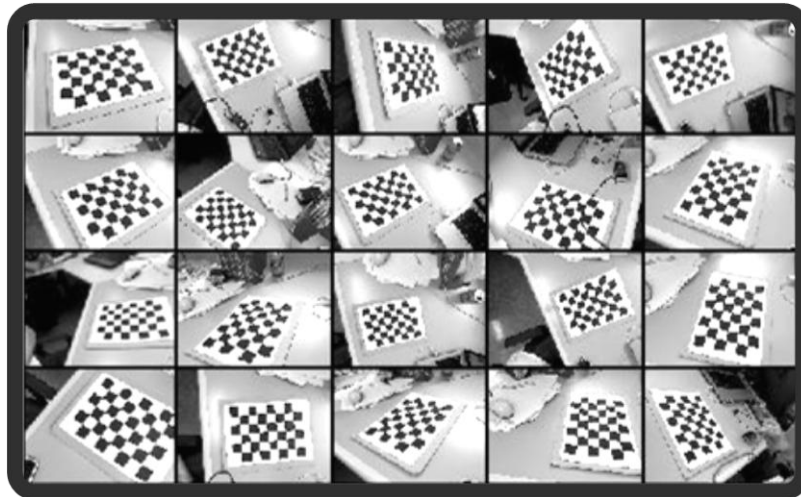
Figura C.1 – Acquisizione delle immagini del pattern di calibrazione nel toolbox per Matlab.

[7] http://www.vision.caltech.edu/bouguetj/calib_doc

Il tempo necessario al calcolo dei parametri della calibrazione risulta notevolmente maggiore rispetto a quello richiesto dal programma sviluppato con OpenCV. Un altro vantaggio, non di minore importanza, a supporto della scelta di implementare una procedura di calibrazione con OpenCV è la possibilità di inserire il codice all'interno di un'applicazione di maggior respiro, quale quella che si ci propone di realizzare con il presente lavoro di Tesi.
Si confrontano, infine, i risultati ottenuti dai due processi di calibrazione.
Sono state acquisite 20 immagini di una scacchiera composta da 6 x 8 riquadri.
Nella tabella seguente (**Tab. C.1**) si confrontano i valori dei parametri intrinseci. Si osserva come entrambi i processi identifichino valori analoghi per le lunghezze focali. Maggiore discordanza dei risultati si ottiene per i valori del punto principale, con una differenza inferiore al 5%.
Le due procedure ottengono risultati molto simili, a prova della bontà del sistema sviluppato.

	OpenCV	Matlab
Lunghezza focale lungo x	550,46	551,29
Lunghezza focale lungo y	550,46	552,60
Punto principale x	315,72	309,92
Punto principale y	240,79	235,97

Tabella C.1 – Confronto dei risultati dei due programmi di calibrazione.
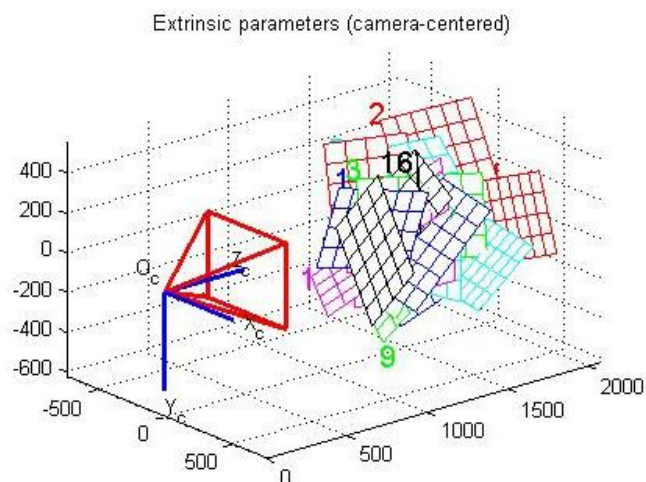
Figura C.2 – Output grafico del programma di calibrazione per Matlab.

Appendice D.
Validazione del sistema di tracciamento

1. Misura delle prestazioni

L'algoritmo di tracciamento, implementato utilizzando le funzioni presenti nella libreria OpenCV e presentate nei capitoli 1 e 4, risulta molto oneroso dal punto di vista computazionale, in modo particolare l'operazione che richiede più tempo è quella del matching dei descrittori, associati alle features estratte dalla scena. Per esempio, su di un frame di dimensioni 1280 x 1024 pixel quest'operazione richiede circa 0.8s, utilizzando un'immagine del marker di dimensioni 210 x 304 pixel.
Le prove effettuate per ridurre il tempo computazionale sono state diverse ed hanno portato ad interessanti risultati, utilizzando differenti strategie.
Il primo approccio è stato quello di ridurre le dimensioni, in pixel, sia dell'immagine catturata con la webcam, che dell'immagine del marker naturale, salvata in memoria, utilizzando la funzione `resize()` di OpenCV
All'aumentare del fattore di scala si riduce, come si ci aspetta, il tempo impiegato nelle fasi di estrazione dei keypoints e di ricerca delle corrispondenze (**Fig. D.1**). Riducendo le dimensioni dell'immagine del marker, utilizzando lo stesso fattore di scala, però, si ha una notevole perdita di informazioni associate all'immagine e una diminuzione della distanza dalla telecamera alla quale il marker viene individuato (**Fig. D.2**).
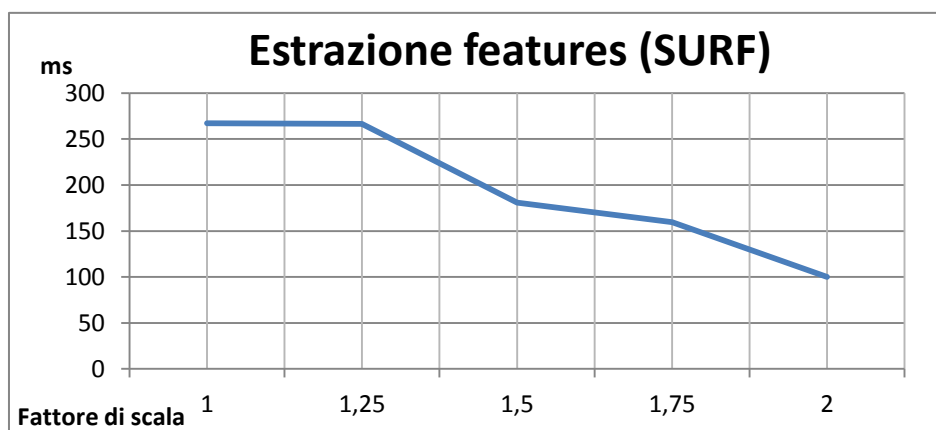
Figura D.1 – Riduzione del tempo computazionale al variare del fattore di scala: fase di estrazione features.
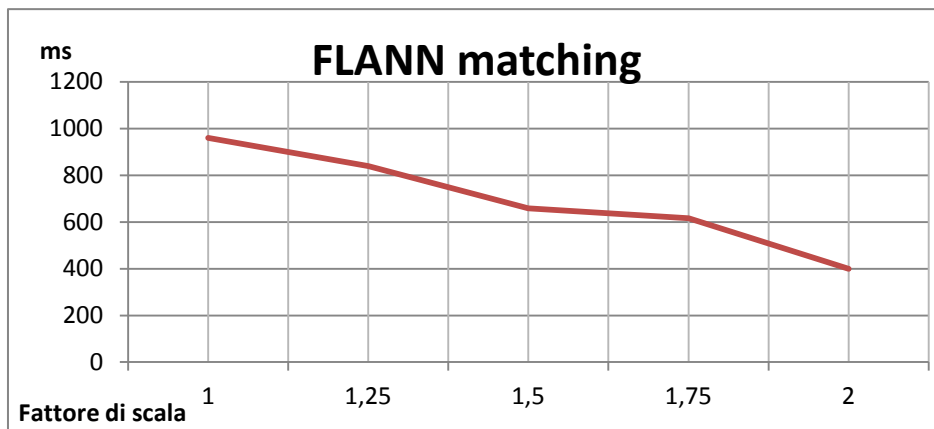
Figura D.2 – Riduzione del tempo computazionale al variare del fattore di scala: fase di matching.

Si è scelto, quindi, di ridimensionare le immagini dei marker per ottimizzarne la ricerca, cercando di intervenire il meno possibile sulla risoluzione dell'immagine, che si vuole visualizzare a 1028 x 720 pixel.
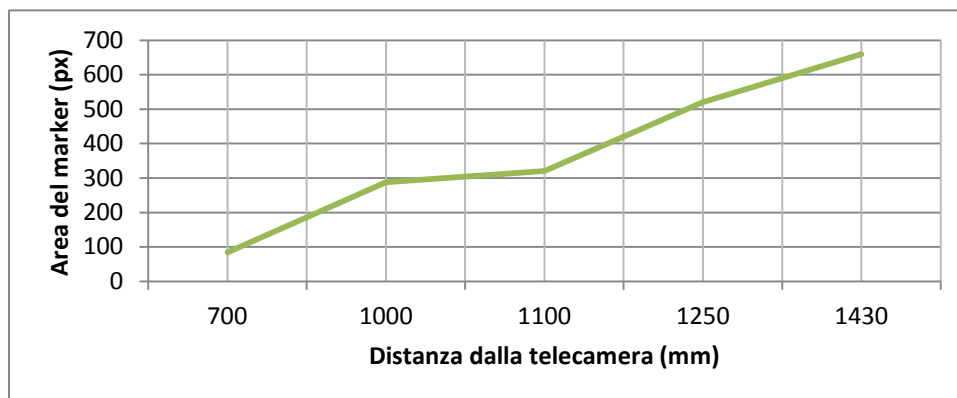
Figura D.3 – Rapporto fra le dimensioni del marker e la distanza alla quale il marker è individuato correttamente ed in modo continuo.

Un secondo parametro sul quale si è cercato di intervenire è il valore di soglia Hessiana da fornire in ingresso all'algoritmo SURF (Capitolo 1).
All'aumentare di tale valore aumenta il numero di keypoints estratti dall'immagine e, con esso, il tempo richiesto per il computo dei descrittori ed il matching.
Dopo aver condotto diverse misure sulle prestazioni dell'algoritmo, si può osservare (**Fig. D.4**) come, all'aumentare del valore di soglia, diminuisca sì il tempo di calcolo ma diminuisca anche il numero di keypoints estratti (**Fig. D.5**). Al diminuire delle features individuate, purtroppo, corrisponde una diminuzione della precisione del nell'individuare correttamente il template, perché le features che l'algoritmo di matching deve elaborare sono maggiormente disperse nell'immagine.
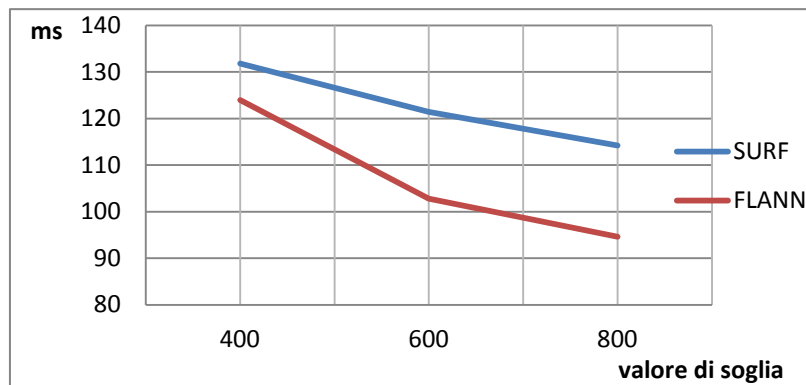
Figura D.4 – Riduzione del tempo di calcolo al variare del valore di soglia Hessiana.

Si è individuato un intervallo di valori della soglia Hessiana, tra 400 e 800, per il quale si raggiunge un buon compromesso fra costi e performance.

Figura D.5 – Keypoints estratti al variare del valore della soglia Hessiana.

L'ultimo intervento, e sicuramente il migliore, compiuto sull'algoritmo è stato quello di inserire una *Region of Interest* (ROI) mobile, all'interno della quale effettuare la ricerca della feature. Tramite l'impiego di una ROI, infatti, è stato possibile ridurre le dimensioni del frame senza perdere in risoluzione.

Come già detto nel paragrafo precedente, la regione di interesse viene definita a partire dalla posizione dell'oggetto trovato nel frame precedente. Appare ovvio che, nel caso in cui non vi siano informazioni su oggetti riconosciuti, la ROI viene a coincidere con l'intera immagine acquisita dalla webcam.
La ROI è definita essere il rettangolo che racchiude l'area del marker riconosciuto, incrementata del 40%, per tenere conto degli spostamenti dell'oggetto o della telecamera (**Fig. D.6**).

Figura D.6 – Estrazione di una Region of Interest, indicata dal rettangolo bianco.

Utilizzando questo espediente il tempo computazionale è stato ridotto a circa 150 ms per l'intero algoritmo, senza perdere in risoluzione e senza dover ridimensionare l'immagine, con la possibilità di impiegare più marker naturali diversi contemporaneamente (**Fig. D.7**).

Figura D.7 – Identificazione di più marker naturali.

2. Validazione delle misure

Prima di passare alla fase dei casi applicativi (capitolo 5) si è scelto di condurre una sessione di test per validare la strategia del tracciamento implementata nel lavoro di Tesi e per valutare l'attendibilità delle coordinate della telecamera su cui si basa in effetti il funzionamento dell'intero sistema.
Si è scelto di eseguire una prima verifica sulla coerenza del sistema in relazione al movimento di una telecamera che inquadra un singolo marker, per valutare inoltre la correttezza dei calcoli per individuare la camera pose.
La seconda verifica del sistema ha portato a validare il sistema di calibrazione del markerfield (capitolo 4) per la creazione di un unico sistema di riferimento assoluto dal quale dipendano tutti i marker.
La prima sessione di test è stata condotta utilizzando diverse immagini di più marker campioni, variando le condizioni di illuminazione dell'ambiente di lavoro.
Fissato il marker in posizione nota rispetto alla telecamera che lo inquadra, si sono acquisite più immagini e si è valutato l'errore medio sulle misure della coordinata z in funzione della distanza. Si è scelto di considerare attendibili tutte le misure che presentassero un errore minore del 6% rispetto alla misura effettiva, acquisita con un metro lineare. Queste prove hanno condotto al grafico in **Fig. D.8**, che mostra come la distanza alla quale le misure siano attendibili dipende dalle dimensioni (in pixel) del marker.

Figura D.8 – Errore sulla coordinata z in funzione della distanza per un marker 65x130 (pixel).

Si è rilevata, inoltre, la necessità di un dato aggiuntivo da utilizzare per calibrare i marker prima dell'acquisizione dei dati, quale ad esempio la dimensione del marker oppure la distanza alla quale viene presa la prima acquisizione, per rafforzare l'accuratezza del tracciamento della *camera pose* che avviene con un sistema monocamera.

Una seconda sessione di test è stata effettuata per valutare la coerenza del tracciamento, individuando prima lo zero del sistema (che corrisponde al centro della telecamera ed al pixel in alto a sinistra dell'immagine del marker) e osservando poi il variare delle misure muovendo la telecamera lungo un solo asse di traslazione alla volta. Si osservano in **Fig. D.9** i grafici che mostrano come, laddove il tracciamento sia ritenuto affidabile per i criteri di cui sopra (errore sulla misure reale inferiore al 6%), la variazione della singola coordinata non ha effetto sulla misura delle altre due.
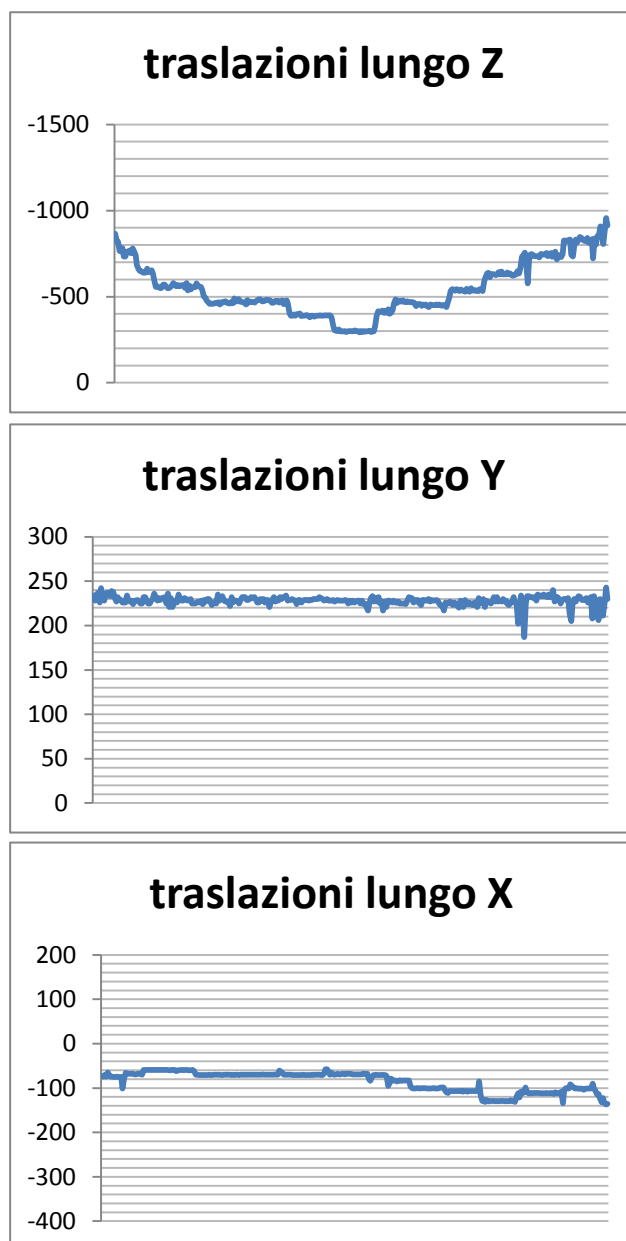
Figura D.9 – Variazione delle coordinate lungo i tre assi al movimento lineare lungo l'asse z della telecamera.

Un altro obiettivo di questa sessione di test è stata la ricerca della massima rotazione del marker lungo i tre assi che consentisse un tracciamento attendibile. Si è osservato che il FOV del singolo marker è di circa 70 gradi (rotazioni massime di 35 gradi lungo gli assi).
Per validare tali risultati si è scelto di visualizzare un parallelepipedo in corrispondenza dello zero del marker e osservare la sua collocazione spaziale al variare della rotazione del marker (**Fig. D.10**).
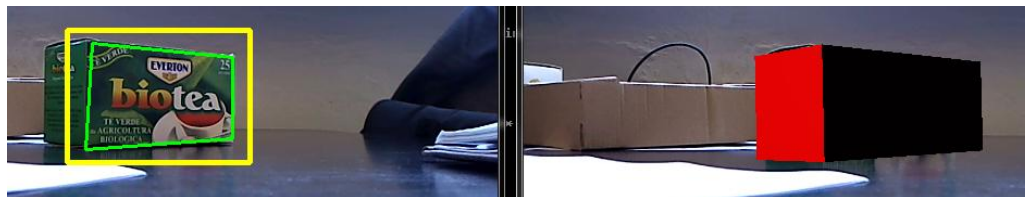
Figura D.10 – Rendering di un parallelepipedo in corrispondenza del marker.

L'ultima serie di test è stata condotta per valutare l'affidabilità del sistema di calibrazione del markerfield, necessario per creare un sistema di riferimento assoluto tra più marker (capitolo 4). Si è scelto di verificare la creazione del markerfield utilizzando, in diversi scenari, coppie di marker, di cui uno è stato scelto come base-marker, origine cioè del sistema di riferimento assoluto. Il test si articola in tre fasi:

- Prima dell'esecuzione si calcola, con l'algoritmo implementato e descritto nel capitolo 4, la matrice di rototraslazione fra i due marker, che rappresenta la trasformazione fra i due sistemi di coordinate (relativo ed assoluto).
- Durante l'esecuzione si calcolano le coordinate della telecamera nei due sistemi di riferimento, utilizzando la tecnica di tracciamento per il singolo marker, verificato nei test precedenti.
- Per ogni frame acquisito, inoltre, si confrontano le coordinate della camera pose valutate sia rispetto al base-marker che rispetto alle coordinate del secondo marker nel sistema di riferimento del base-marker.

Gli scenari in cui si sono effettuati i test sono stati tre:

- Marker complanari (**Fig. D.11**)
- Marker paralleli ma non complanari (**Fig. D.12**)
- Marker disposti su piani incidenti (**Fig. D.13**)

Dall'analisi dei risultati (**Fig. D.14-15-16**) si osserva che le misure effettuate con la trasformazione di coordinate sono coerenti con quelle del tracciamento per il singolo marker e, in alcuni casi, ne riducono il rumore.

Figura D.11 – Marker complanari.
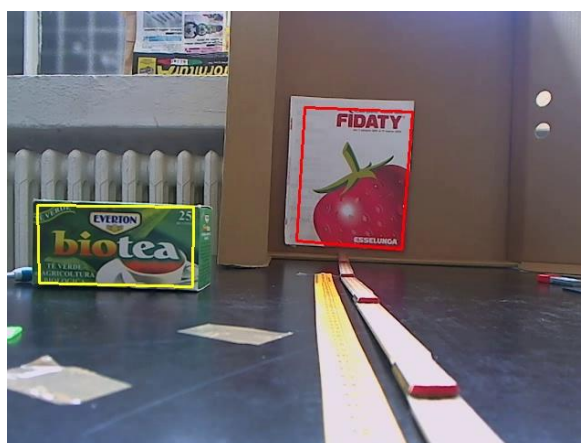
Figura D.12 – Marker paralleli.

Figura D.13 – Marker posti su piani incidenti.
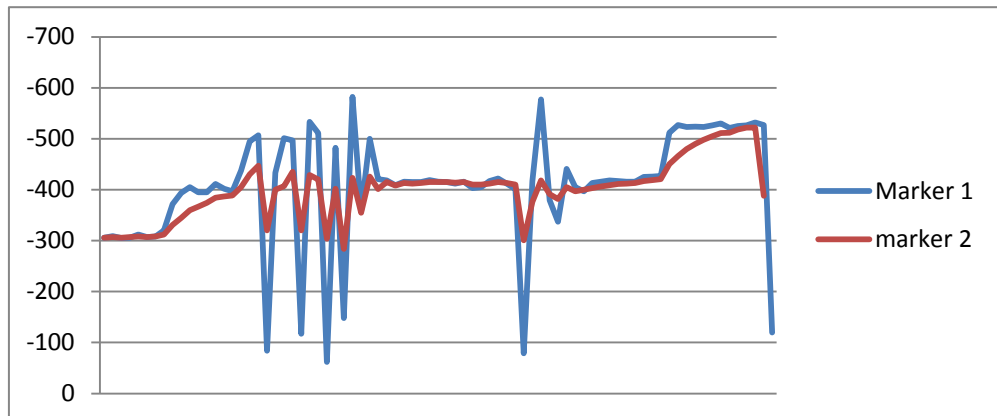
Figura D.14 – Risultati configurazione 1. In blu le coordinate calcolate nel riferimento del base-marker, in rosso quelle calcolate con la matrice di trasformazione.
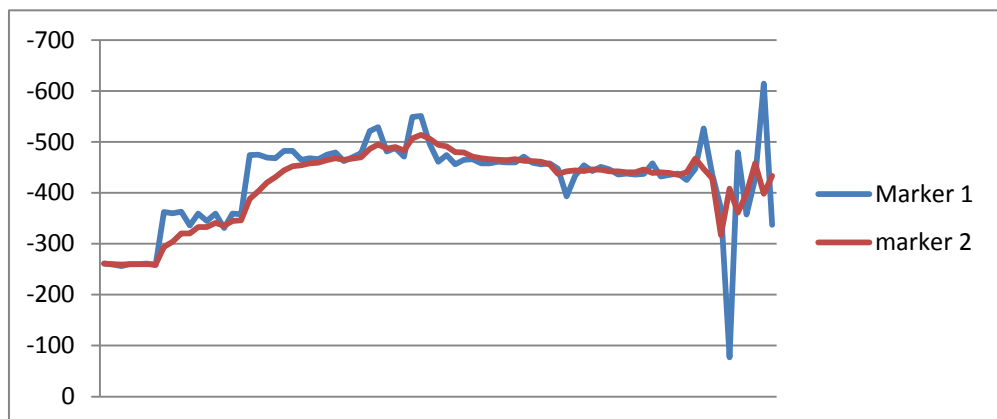
Figura D.15 – Risultati configurazione 2. In blu le coordinate calcolate nel riferimento del base-marker, in rosso quelle calcolate con la matrice di trasformazione.
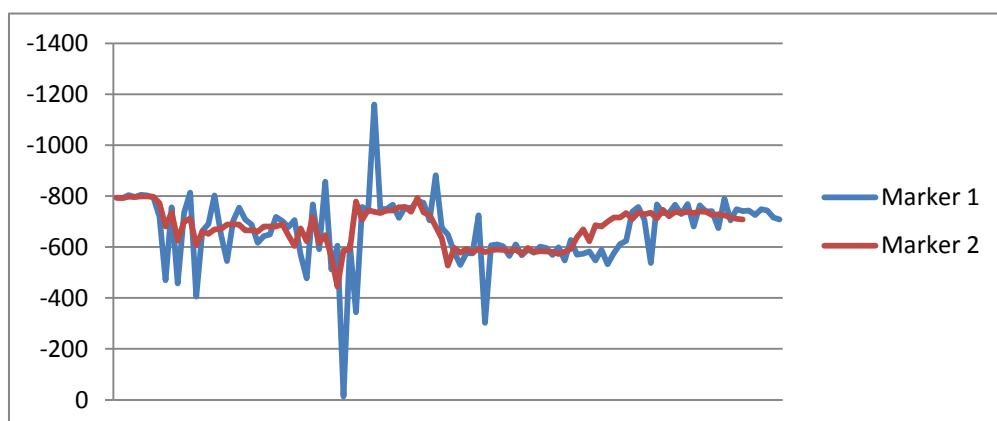
Figura D.16 – Risultati configurazione 3. In blu le coordinate calcolate nel riferimento del base-marker, in rosso quelle calcolate con la matrice di trasformazione di coordinate.

Bibliografia

[1] Alben L., *Quality of experience: defining the criteria for effective interaction design,* "I*nteractions", Vol.*3, No.3, May 1996, pp.11-15.

[2] Allison R. S., Harris L. R., Jenkin M., *Tolerance of temporal delay in virtual environments,* in *"*Proceedings of IEEE Virtual Reality 2001", Yokohama, March 2001, pp. 247-254.

[3] Azuma R. T., *A survey of augmented reality*, "Teleoperators and Virtual Environments", Vol. 6, No. 4, August 1997, pp. 355-385.

[4] Azuma R., Baillot Y., Behringer R., Feiner S., Julier S., *Recent advances in Augmented reality*, "Computer Graphics and Applications", Vol. 21, No.6, November 2001, pp. 34-47.

[5] Azuma, R., Daily M., Krozel J., *Advanced human computer interfaces for air traffic management and* Simulation, in "Proceedings of AIAA Flight Simulation Technologies Conference", San Diego, July 1996, pp.656-666.

[6] Baird K., Barfield W., *Evaluating the effectiveness of augmented reality displays for a manual assembly task*, "Virtual Reality", Vol. 4, No. 4, December 1999, pp. 250-259.

[7] Baratoff G., Neubeck A., Regenbrecht H., *Interactive multi-marker calibration for augmented reality applications*, in "Proceedings of ISMAR 2002: International Symposium on Mixed and Augmented Reality", Darmstadt, September 2002, pp. 104-109.

[8] Barfield W., Caudell T., *Fundamentals of Wearable Computers and Augumented Reality*. L. Erlbaum Associates Inc., Hillsdale 2000.

[9] Bay H., Ess A., Tuytelaars T., Van Gool L., *Speeded-Up Robust Feature (SURF)*, in "ECCV", Vol. 110, No. 3, June 2008, pp. 346-369.

[10] Beauchemin S. S., Barron J. L., *The computation of optical flow*, "ACM Computing Surveys", Vol. 27, No. 3, September 1995, pp. 433-466.

[11] Boothroyd G., Dewhurst P., Knight W., *Product Design for Manufacture and Assembly*, Marcel Dekker, New York 2002.

[12] Bradski G., Kaehler A., *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media Inc., Sebastopol 2008.

[13] Calonder M., Lepetit V., Strecha C., Fua P., *BRIEF: Binary Robust Independent Elementary Features*, in "Proceedings of European Conference on Computer Vision (ECCV)", Berlin, September 2010, pp. 778-792.

[14] Caudell T. P., Mizel*l* D.W., *Augmented reality: an application of heads-up display technology to manual manufacturing processes*, in "Proceedings of Hawaii International Conference on System Sciences", Kauai, January 1992, Vol. 2, pp. 659-669.

[15] Fadini A., *Tecniche di visione artificiale e realtà aumentata finalizzate all'assistenza di un operatore durante l'attrezzaggio di una macchina utensile*. Tesi di Laurea Magistrale, Politecnico di Milano 2011.

[16] Farin D., *Automatic Video Segmentation Employing Object/Camera Modeling Techniques*. PhD Thesis, Eindhoven University 2005.

[17] Feiner S., Macintyre B., Seligmann D., *Knowledge-based augmented reality*, "Communication of the ACM", Vol. 36, No. 7, July 1993, pp. 52-62.

[18] Fischler M. A., Bolles R. C., *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, "Communications of ACM", Vol. 24, No.6, June 1981, pp. 381-395.

[19] Friedmann M., Starner T., Pentland A., *Synchronization in virtual realities*, "Presence: Teleoperators and Virtual Environments", Vol. 1, No.1, March 1992, pp. 139-144.

[20] Friedrich W., *ARVIKA-augmented reality for development, production and service* , in "Proceedings of International Symposium on Mixed and Augmented Reality", Darmstadt, September 2002, pp. 3-4.

[21] Gao T., Packer B., Koller D., *A Segmentation-aware Object Detection Model with Occlusion Handling*, in "Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)", Colorado Springs, June 2011, pp. 1361-1368.

[22] Gil A., Mozos O., Ballesta M., Reinoso O., *A comparative evaluation of interest point detectors and local descriptors for visual SLAM*, "Machine Vision and Applications", No. 21, November 2009, pp. 905-920.

[23] Graves M. , Batchelor B. G. *Machine vision for the inspection of natural products*, Springer, Berlin 2003.

[24] Haberland, U., Brecher, C., Possel-Dlken, F., *Advanced augmented reality-based service technologies for production systems*, in "Proceedings of the International Conference on Smart Machining Systems", Gaithersburg, March 2007, pp. 31-35.

[25] Hartley R., Zisserman A., *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge 2003, 2nd edition.

[26] Henderson S., Feiner S., *Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret*, in "Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)", Orlando, October 2009, pp. 135-144.

[27] Hinterstoisser S., Cagniart C., Ilic S., Lepetit V., *Gradient Response Maps for Real-Time Detection of Texture-Less Objects*, in "IEEE Transactions on Pattern Analysis and Maschine Intelligence (TPAMI)",Vol. 34, No.5, May 2012, pp. 1281-1298.

[28] Hinterstoisser S., Lepetit V., Ilic S. *Dominant orientation templates for real-time detection of texture-less objects*, in "Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition", San Francisco, June 2010, pp. 106 -112.

[29] Hollerer T., Feine S., *Mobile Augmented Reality*, Francis Books Ltd., London 2004.

[30] Jacobs M., Livingston A., State A., *Managing latency in complex augmented reality systems*, in "Proceedings of the 1997 symposium on Interactive 3D graphics", Providence, April 1997, pp. 49-54.

[31] Kato H., Billinghurst M., *Marker tracking and HMD Calibration for a videobased augmented reality conferencing system*, in "Proceedings of International Workshop on Augmented Reality", San Francisco, October 1999, pp. 85-94.

[32] Kato H., Billinghurst M., Poupyrev I., Imamoto K., *Virtual object manipulation on a table-top AR environment*, in "Proceedings of International Symposium on Augmented Reality", Munich, October 2000, pp. 111-119.

[33] Kleiber M., Alexander T., *Evaluation of a Mobile AR Tele-Maintenance System*, in "Universal Access in Human-Computer Interaction. Applications and Services", Springer, Berlin 2011, pp 253-262.

[34] Klein G., Murray D., *Parallel Tracking and Mapping for Small AR Workspaces*, *Mixed and Augmented Reality,* in "Proceedings of International Symposium on Mixed and Augmented Reality", Nara, November 2007, pp. 1-10.

[35] Koller G., Klinker E., Rose D., Breen R., *Realtime vision-based camera tracking for augmented reality applications*, in "Proceedings of the ACM Symposium on Virtual reality software and technology", Losanna, September 1997,pp. 87-94.

[36] Konolige K., Agrawal M., *Rough terrain visual odometry*, in "Proceedings of International Conference on Advanced Robotics (ICAR)", Jeju Island, August 2007, pp. 56-62.

[37] Kumar P., Koller D., *Efficiently Selecting Regions for Scene Understanding*, *in "*Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)", San Francisco, June 2010, pp. 21-29.

[38] Leigh J., Vasilakis C., DeFanti T., *Virtual reality in computational neuroscience*, in "Proceedings of the Conference on Applications of Virtual Reality", British Computer Society, Singapore, August 1994, pp 293-306.

[39] Lindeberg T., Bretzner L., *Real-time scale selection in hybrid multiscale rappresentations*, in "Proceedings of Scale-Space Methods in Computer Vision", Isle of Skye, June 2003, pp. 148-163, 2003.

[40] Lindeberg T., *Scale-space for discrete signals*, "IEEE Transactions of Pattern Analysis and Machine Intelligence", Vol. 12, No. 3, January 1990, pp. 234-254.

[41] Lowe D. *Distinctive image features from scale-invariant keypoints, cascade filtering approach*, "International Journal of Computer Vision", Vol. 60. No. 2, January 2004, pp. 91-110.

[42] Lowe D., Se S., Little J., *Vision-based Mobile ROBOT Localization and Mapping using Scale-Invariant Features*, in "Proceedings of the IEEE International Conference on Robotics & Automation", Seoul, May 2001, pp. 2051-58.

[43] Lucas B. D., Kanade T., *An Iterative Image Registration Technique with an Application to Stereo Vision*, in "Proceedings of Imaging Understanding Workshop", Washington, April 1981, pp. 121-130.

[44] Marr D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Freeman and Company, San Francisco 1982.

[45] Mayez A., *Introduction to Robot Programming.* Computer Engineering Department, King Fahd University of petroleum and Minerals, Dhahran 2008.

[46] Meehan M., Razzaque S., Whitton M., *Effect of Latency on Presence in stressful virtual environments*, in "Proceedings of the IEEE Virtual Reality", Los Angeles, March 2003, pp. 155-162.

[47] Meier P., Platonov J., Heibel H., Grollmann B., *A mobile markerless AR system for maintenance and repair*, in "Proceedings of Fifth International Symposium on Mixed and Augmented Reality", Santa Barbara, October 2006, pp. 105-108.

[48] Michalak D., *Applying the Augmented Reality and RFID Technologies in the Maintenance of Mining Machines*, in "Proceedings of the World Congress on Engineering and Computer Science 2012", Vol. 1, San Francisco, October 2012, pp.16-21.

[49] Milgram P., Takemura H., Utsumi A., *Augmented reality: A class of displays on reality-virtuality continuum*, in "Proceedings of the SPIE Conference on Telemanipulator and Telepresence Technologies", Munich, November 1995, Vol. 2351, pp. 282-292.

[50] Muja M., Lowe, D. *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration*, in "Proceedings of *International Conference on Computer Vision Theory and Applications (VISAPP)",* Lisbon, February 2009, pp. 331-340.

[51] Mulder A., *Human movement tracking technology*. Technical report 94-1, School of Kinesiology, Simon Fraser University, Vancouver, July 1994.

[52] Newman J., Fraundorfer F., Schall G., Schmalstieg D. *Construction and maintenance of augmented reality environments using a mixture of autonomous and manual surveying techniques,* in "Proceedings of the 7th Conference on Optical 3-D Measurement Techniques", Vienna, October 2005, pp.228-234.

[53] Reiners D., Stricker D., Klinker G., *Augmented reality for construction tasks: Doorlock assembly*, in "Proceedings of IEEE International Workshop on Augmented Reality", Reno, March 2008, pp. 282-290.

[54] Rublee E., Rabaud V., Bradski G., *ORB: An Efficient Alternative to SIFT or SURF*, in "Proceedings of International Conference on Computer Vision", Barcelona, November 2011, pp. 24-30.

[55] Schall G., Schoening J., Paelke V., *A survey on augmented maps and environments: Approaches, interactions and applications*, in "Advances in Web-based GIS, Mapping Services and Applications", Taylor and Francis Group, London 2011.

[56] Schweighofer G., Pinz A., *Robust Pose Estimation from a Planar Target*, "IEEE Transactions on Pattern Analysis and Machine Intelligence", Vol.28, No.12 (December 2006), pp.2024-2030.

[57] Shapiro L.G., Stockman G. C. *Computer Vision*. Prentice Hall, Boston 2001.

[58] Shim J., Warkentin M., Courtney J., Power D., *Past, present, and future of decision support technology*, “Decision Support Systems”, Vol.33, No. 1 (May 2002), pp. 111-126.

[59] Shreiner D., Sellers G., Kessenich J., *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, versions 4.1, Addison-Wesley, Boston 2012.

[60] Soro A., et al., *Human Computer Interaction. Fondamenti e Prospettive*, Polimetrica Publisher, Roma 2008.

[61] Szeliski R., *Computer Vision: Algorithms and Applications*, Springer, Berlin 2010.

[62] Tatikonda M., *Design for Assembly: A Critical Methodology for Product Reengineering and new Product Development,* “Production Inventory Manage. J.”, Vol. 35, No. 1, May 1994, pp. 31-38.

[63] Tausel M., *Appunti del corso di Computer Vision*, Politecnico di Milano, 2011.

[64] Tomasi, C., Kanade, T., *Detection and Tracking of Point Features,* Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[65] Trucco E., Verri A.: *Introductory Techniques for 3D Computer Vision*, Prentice-Hall, Boston 1998.

[66] Viksten F., Forssén P., Johansson B., *Comparison of Local Image Descriptors for Full 6 Degree-of-Freedom Pose Estimation*, in “Proceedings of 2009 IEEE International Conference on Robotics and Automation”, Tokyo, May 2009.

[67] Wahl F., Thomas U., *Robot Programming - From Simple Moves to Complex Robot Tasks,* in “Proceedings of the First International Colloquium 'Collaborative Research Centre 562 - Robotic Systems for Modelling and Assembly' ”, Braunschweig, May 2002, pp. 245-259.

[68] Williams B., *Interaction-based invention: Designing novel devices from first principles,* in Gottlob G., Nejdl W. (eds.), *Expert Systems in Engineering Principles and Applications*, Springer-Verlag, Vienna 1990, pp. 119-134.

[69] Wohlhart P., Roth P., Bischof H., *3D Camera Tracking in Unknown Environments by On-line Keypoint Learning*, “Computer Vision Winter Workshop”, Prague, February 2010.

[70] Y. M. Wang, Y. Li, and J. B. Zheng, *A Camera Calibration Technique Based on OpenCV*, in “Proceedings of Information Sciences and Interaction Sciences (ICIS)”, Saint Louis, December 2010, pp.403-406.

[71] Yuko U., Hideo S. *A registration by merging multiple planar markers at arbitrary positions and poses via projective space,* in “Proceedings of the 2005 International Conference on Augmented Tele-existence”, Christchurch, December 2005, pp. 48-55.

[72] Zhang Z., *A flexible camera calibration by viewing a plane from unknown orientations*, in “Proceedings of the International Conference on Computer Vision”, Corfu, September 1999, pp. 666-673.

[73] Zhang Z., *A flexible new technique for camera calibration*, “On Pattern Analysis and Machine Intelligence”, Vol. 22, November 2000, pp.1330-1334.

[74] Zlatanova S. *Augmented Reality Technology*, “GIST Report No.17”, Delft, December 2002.

Sitografia

[75] http://graphics.cs.columbia.edu/projects/armar/index.htm
[76] http://it.playstation.com/psp/games/detail/item229702/The-Eye-of-Judgement-Portable/
[77] http://manuvar.eu/
[78] http://virtual.vtt.fi/virtual/proj2/multimedia/projects/plamos_brochure.pdf
[79] http://vrmedia.it/it/augmented-reality.html
[80] http://wearables.unisa.edu.au/projects/arquake/
[81] http://www.aec2000.it/archeoguide/
[82] http://www.artesas.de/
[83] http://www.arvika.de/
[84] http://www.hitl.washington.edu/artoolkit/
[85] http://www.joinpad.net/2011/04/11/augmented-xp-2/
[86] http://www.lucense.it/content.php?p=2.1.1.1.4
[87] http://www.media-trust.ru/Catalog/Toshiba/toshibam50h-i.pdf
[88] http://www.metaio.com/
[89] http://www.naturalpoint.com/optitrack/
[90] http://www.ultimate3dheaven.com/noname2.html
[91] http://www.vision.caltech.edu/bouguetj/calib_doc
[92] http://www.vtt.fi/files/research/ism/manufacturingsystems/augasse_flyer.pdf
[93] http://opencv.org/
[94] http://www.opengl.org/
[95] http://campar.in.tum.de/personal/hinterst/index/index.html